University of Isfahan
Department of Mathematical Sciences

From Archetypes to Algorithms: A Neuro-Symbolic Architecture for Translating Literary Formulas into Coherent Narrative Generation

Author: Maryam Rezaee **Student ID:** 981813088
Course: Algorithms & Data Structures 1399-1

14 Jan 2021 | Edited Jul 2025

Abstract

Human storytelling is not random; it relies on time-tested structures that guide the audience from a beginning to a meaningful conclusion. In contrast, current Artificial Intelligence, particularly Large Language Models (LLMs), faces a significant "horizon problem": while they are statistically fluent at the sentence level, they frequently fail to maintain global narrative coherence, resulting in "wandering plots" that lack long-term direction. This paper proposes a solution by reframing literary theory not as artistic convention, but as a cognitive algorithm. We demonstrate that canonical frameworks—such as Freytag's Pyramid and the Hero's Journey—can be operationalised into formal computational paradigms like Signal Processing and Graph Theory. By viewing these structures as efficient data compression schemas, we argue they are essential for minimising predictive error in the brain. Finally, we propose a Neuro-Symbolic architecture that combines the creative flexibility of neural networks with the logical rigidity of these symbolic structures, offering a path towards automated storytelling that is both inventive and logically consistent.

1 Introduction: The Algorithm of Story

From a cognitive standpoint, the human brain is not merely a passive receiver of information; it is a prediction machine. When confronted with the high-entropy chaos of sensory reality—where anything can happen at any moment—the brain constantly seeks to minimise uncertainty. In neuroscience, this is known as minimising "surprisal" [1]. This research posits that storytelling is not merely an artistic endeavour, but a sophisticated **Data Compression Algorithm** evolved to facilitate this predictive processing. In other words, a story compresses the overwhelming complexity of social reality into a structured format with clear, causal patterns, serving as a cognitive heuristic for transforming overwhelming sensory data into manageable, memorable units [2].

Traditionally, the study of narrative has been the domain of the humanities, analysed through qualitative literary criticism. However, when viewed through the lens of Computer Science, the trans-cultural persistence of specific narrative structures—such as the Monomyth or the Three-Act Structure—suggests something rather interesting. If these patterns appear independently across human history, they are likely efficient, optimised **Data Structures** for encoding causal information.

This research aims to formalise the "Soft Science" of narratology into the "Hard Science" of algorithmic modelling. By deconstructing literary theory through the lens of **Advanced Algorithms**, we aim to answer a critical question in modern Artificial Intelligence: *How can we move beyond the "stochastic parroting" of current Neural Language Models to create systems that actually plan, reason, and tell coherent stories?* Current AI is excellent at predicting the next word, but it often struggles to predict the next plot point.

This paper is structured to bridge the gap between literary theory and computational implementation:

- **Section 2:** We map seven canonical narrative structures to formal algorithmic paradigms. We demonstrate how literary arcs can be modelled using Signal Processing, Graph Theory, and Constraint Satisfaction Problems (CSP).
- **Section 3:** We move from plot to character. We quantify character archetypes not as mystical symbols, but as **Vector Centroids** in a high-dimensional latent space, allowing for mathematical consistency in character generation.
- **Section 4:** We ground these computational models in cognitive neuroscience. We argue that these structures exist to satisfy the Free Energy Principle, serving as tools to minimise informational entropy in the human mind.
- **Section 5:** Finally, we propose a novel **Neuro-Symbolic Architecture**. This system integrates the logic of symbolic planning with the creativity of neural generation to solve the "Wandering Plot" problem inherent in models like GPT-3.

By treating narrative not as magic, but as a search problem within a state space, we demonstrate that the path to robust Artificial Intelligence lies in the synthesis of statistical learning and symbolic structuralism.

2 Algorithmic Deconstruction of Narrative Structures

Building upon the cognitive modelling objectives outlined in the previous section, we now turn our attention to the most observable artefact of human thought: the structure of narrative. Before we can effectively model complex generative cognition, we must first operationalise the structural blueprints humans use to organise and communicate experience. Consequently, the primary domain of our technical exploration is **Narrative Structure**.

To render these literary concepts computationally viable, we utilise the framework of *Computational Narratology*. This field bridges the gap between the humanities and formal logic, translating the qualitative rules of storytelling into quantitative schemas that a computer can process [3]. Effectively, this allows us to transpose the heuristics of literary theory into the rigorous constraints of computer science.

For artificial intelligence, particularly large language models, a significant challenge is the absence of intrinsic intent. As probability-based systems, these models often produce a "wandering plot"—narratives that maintain local coherence but lack global direction. To mitigate this, we propose treating narrative structures as **Cost Functions**.

In the context of optimisation theory, a cost function serves as a metric for quantifying error—specifically, the discrepancy between a system's current state and its ideal target state. For a language model, which probabilistically predicts the next word, there is typically no inherent concept of a "bad" plot twist, only a "statistically unlikely" word sequence. By encoding narrative rules as cost functions, we introduce an external standard of validity. If the model generates a text segment that deviates from the structural blueprint (e.g., generating a calm scene when the structure demands a climax), the function returns a high "cost" value. The generative process is thus transformed into an objective-based search, where the system strives to traverse the path of minimal narrative error.

We classify these algorithmic approaches into three distinct mathematical paradigms:

- **Continuous Models (Signal Processing):** Utilised for narratives defined by the modulation of rising and falling tension (e.g., Freytag's Pyramid). Analogous to a feedback control loop (such as a thermostat), these systems continuously adjust the narrative intensity to track a specific reference curve over time.
- **Discrete Models (State Machines):** Utilised for narratives dependent on logical prerequisites and inventory states (e.g., The Hero's Journey). This is functionally similar to game design logic, where a character's progression to the next level is gated by the acquisition of specific items or the completion of prerequisite tasks.
- **Search Heuristics (A*):** Utilised for narratives governed by rigid temporal or pacing constraints (e.g., Save the Cat). These operate on principles comparable to navigation pathfinding, where the algorithm must identify the most efficient route to a destination while strictly adhering to a limitation on time or length.

In the following subsections, we deconstruct seven canonical storytelling frameworks, translating them from literary theory into formal algorithms.

2.1 Narrative Structure: Freytag's Pyramid

2.1.1 History and Theoretical Framework

We begin with the fundamental model of Western dramatic structure. Based on an analysis of classical Greek tragedy and Shakespearean drama, Gustav Freytag proposed that effective narratives follow a pyramidal shape of tension [4]. He posited that a story is not a linear progression of events, but a precise elevation and release of conflict. This symmetrical distribution of tension is illustrated in Figure 1.

While originally descriptive of tragedy, this model essentially functions as a schematic for managing cognitive load. The structure moves from low entropy (order), through high entropy (chaos), and back to order. The five distinct movements are:

1. **Introduction (Exposition):** The establishment of the status quo. The setting and characters are introduced prior to the introduction of conflict.
2. **Rise (Rising Action):** The inciting incident disrupts the equilibrium. The protagonist encounters complications, and narrative tension increases monotonically.
3. **Climax (Turning Point):** The apex of the pyramid. Freytag defined this as the "point of no return," determining the protagonist's ultimate fate.
4. **Return/Fall (Falling Action):** The immediate aftermath. The narrative unravels the consequences of the climax.
5. **Catastrophe (Denouement):** The final resolution. The conflict is resolved, and the narrative state stabilises.

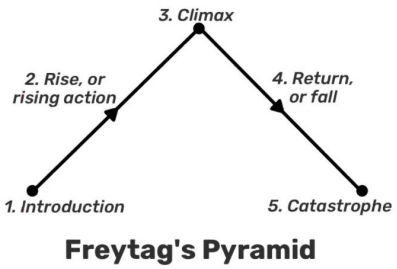

Figure 1: Freytag's Pyramid: A signal processing envelope for narrative tension, showing the symmetrical rise and fall of conflict intensity.

2.1.2 Algorithmic Implementation

Computationally, we model Freytag's Pyramid not as a static template, but as a dynamic **Feedback Control Loop** [5]. In control systems engineering, a feedback loop continually monitors a system's output and adjusts the input to minimise error. Applied to narrative generation, this mechanism functions analogously to a thermostat or an automated volume regulator: it actively maintains the story's emotional "tempera-

ture" by constantly comparing the current text against a desired tension level. In this framework:

- The **Signal** is the "Narrative Tension." We measure this by calculating the *Sentiment Inverse* of the text (where negative sentiment, such as fear or anger, equates to high tension).
- The **Setpoint** is the Gaussian Curve $V_{\text{target}}(t)$. This serves as the reference trajectory, dictating exactly how intense the scene *should* be at any specific timestamp t.

Vocabulary Injection. The core mechanism for correcting the narrative flow is **Token Injection** via Logit Bias. Large Language Models predict the next word by assigning probabilities to their entire vocabulary. To steer the story, we intervene in this probability layer based on the calculated error $\Delta = V_{\text{target}}(t) - V_{\text{current}}$.

1. **Scenario A: Under-Tensioned ($\Delta > 0$).** If the system detects the story is too calm during the "Rising Action" phase, it activates a dictionary of **Complication Tokens**. It artificially boosts the probability of high-arousal verbs and adjectives.
   - *Injected Lexicon:* {"Scream", "Suddenly", "Break", "Enemy", "Fail", "Dark"}
   - *Result:* The model is mathematically forced to generate a sentence containing conflict, e.g., *"Suddenly, the glass broke."*
2. **Scenario B: Over-Tensioned ($\Delta < 0$).** If the system detects high conflict during the "Falling Action" (where the story should be resolving), it activates **Resolution Tokens**.
   - *Injected Lexicon:* {"Breath", "Silence", "Safe", "Understood", "Calm", "Light"}
   - *Result:* The model is steered towards de-escalation, e.g., *"She took a deep breath and the room fell silent."*

This ensures the generated text tightly tracks the predefined Gaussian curve, as detailed in Algorithm 1.

Algorithm 1: Freytag Signal Generation

Input: TargetLength L, PeakTime T_{peak}, Params θ

$t \leftarrow 0$, $CurrentValence \leftarrow 0$;
$S \leftarrow$ GenerateExposition();
while $t < L$ **do**
  // Calculate Target from Gaussian Eq. 1
  $V_{\text{target}} \leftarrow A \cdot \exp\left(-\frac{(t-T_{\text{peak}})^2}{2\sigma^2}\right)$;
  $CurrentValence \leftarrow$ AnalyseSentiment($S_{\text{last\_scene}}$);
  // Calculate Error Delta
  $\Delta \leftarrow V_{\text{target}} - CurrentValence$;
  if $\Delta > \delta$ **then**
    // Under-tensioned: Boost Conflict
    InjectLogits(ConflictLexicon, Strength = Δ);
  end
  else if $\Delta < -\delta$ **then**
    // Over-tensioned: Boost Relief
    InjectLogits(ReliefLexicon, Strength = $|\Delta|$);
  end
  $Event \leftarrow$ GenerateScene();
  S.append($Event$);
  $t \leftarrow t + 1$;
end

2.1.3 Mathematical Modelling Proposition: Gaussian Trajectories and Tension Envelopes

To strictly enforce the rising and falling action described by Freytag, we model the narrative arc as a continuous function of time. A significant issue in generated text is the lack of tonal consistency, where a story might jump abruptly from mundane to catastrophic without buildup. To prevent this, we utilise a modified Gaussian function (Bell Curve). This provides a smooth, differentiable envelope for tension, ensuring that conflict swells organically rather than spiking instantaneously.

Let $V_{\text{target}}(t)$ represent the ideal tension value at normalized time $t \in [0, 1]$:

$$V_{\text{target}}(t) = A \cdot \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right) + \epsilon \tag{1}$$

In this equation, the variables act as narrative "dials," allowing us to precisely shape the story's emotional profile:

- **Amplitude (A):** Represents the *Stakes*. A high A indicates a high-intensity thriller; a low A indicates a slice-of-life drama.
- **Mean (μ):** Represents the *Pacing/Timing* of the Climax. In a standard Freytag structure, this is typically skewed to the right (e.g., $\mu \approx 0.8$), ensuring a long rising action and a rapid denouement.
- **Standard Deviation (σ):** Represents the *Duration* of the conflict. A narrow σ creates a sudden, shocking climax; a wide σ creates a drawn-out, epic confrontation.
- **Bias (ϵ):** The baseline tension (to prevent total apathy).

Optimisation via Loss Function. The AI utilises this function as a comparator. For every generated text segment S_t, we calculate its semantic sentiment intensity, denoted as $V_{\text{gen}}(S_t)$, using a pre-trained sentiment analyser (e.g., VADER or BERT). We then calculate the squared error loss $\mathscr{L}$:

$$\mathscr{L}(t) = \left(V_{\text{gen}}(S_t) - V_{\text{target}}(t)\right)^2 \tag{2}$$

If the AI generates a low-tension scene (e.g., a joke) at time $t = \mu$ (the Climax), $\mathscr{L}(t)$ yields a high value. This error signal acts as a negative reward, prompting the model to discard the segment and regenerate the sequence with higher conflict intensity until $\mathscr{L}(t) < \delta$ (where δ is an acceptable threshold).

2.2 Narrative Structure: The Fichtean Curve

2.2.1 History and Theoretical Framework

While Freytag models classical tragedy, the Fichtean Curve, popularized by John Gardner [6], is optimised for high-tension modern fiction such as thrillers. This structure bypasses extended exposition, immersing the reader immediately in an "Inciting Incident."

Visually, the structure resembles a staircase, as depicted in Figure 2. Unlike Freytag's single rise, the Fichtean model is iterative. The protagonist encounters a crisis, resolves it, but the resolution itself triggers a new, more dangerous set of circumstances. This creates a ratchet effect: the hero cannot return to the previous state of safety, and the "price" of failure increases with every step.

1. **Inciting Incident:** A disruptive event occurring immediately at the narrative onset.
2. **Rising Action (Series of Crises):** A recursive loop of conflict. Each crisis creates higher stakes than the predecessor.
3. **Climax:** The absolute peak of tension.
4. **Falling Action:** A minimal resolution phase.

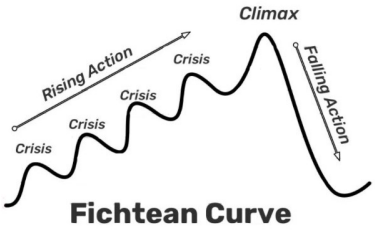

Figure 2: The Fichtean Curve: Conflict accumulation modelled as a series of increasing local maxima leading to a global maximum.

Algorithm 2: Fichtean Iterative Generation

Input: MaxDifficulty $D_{\max}$, InitialState S_0

$S \leftarrow$ IncitingIncident(S_0);
$Difficulty \leftarrow 1$;
while $Difficulty < D_{max}$ **do**
    `// Run the 'Crisis' subroutine loop`
    $CrisisState \leftarrow$ CreateObstacle($Difficulty$);
    $Resolved \leftarrow$ **False**;
    while *not* $Resolved$ **do**
        $Attempt \leftarrow$ AgentAction($CrisisState$);
        if $CheckSuccess(Attempt, Difficulty)$ **then**
            $Resolved \leftarrow$ **True**;
            UpdateNarrative($Success$);
        end
        else
            UpdateNarrative($Failure$);
            $Difficulty \leftarrow Difficulty \times 1.5$ `// Failure escalates stakes`
        end
    end
    `// Escalate for next cycle`
    $Difficulty \leftarrow Difficulty + 1$;
end
ExecuteGlobalClimax();

2.2.2 Algorithmic Implementation

From a computer science perspective, the Fichtean Curve is best modelled using the principle of **Iterative Deepening** or a **Compound Loop**. This structural approach mirrors level design in video games: the protagonist (player) faces a challenge (boss), and overcoming it does not end the game but merely unlocks the next, more difficult level. The central goal is decomposed into self-similar sub-problems, where the solution of one crisis creates the preconditions for a more difficult subsequent crisis.

The algorithm uses the iterative `While` loop (Algorithm 2). We introduce a global variable `Difficulty` which functions as a scalar multiplier for the conflict generation. The system relies on a permanent rise in stakes: failure immediately compounds the difficulty (×1.5), representing a "disaster" scenario where the hero is worse off than before. Even success increments the difficulty (+1), representing the "yes, but..." trope where solving one problem reveals a larger one. This prevents the system from resolving the narrative before hitting `MaxDifficulty` (the Climax).

2.2.3 Mathematical Modelling Proposition: Superposition of Gaussian Pulses

To mathematically replicate the "staircase" structure of the Fichtean Curve, we model the narrative tension $V(t)$ not as a wave, but as a linear baseline superimposed with a series of discrete impulse events. We utilise a summation of **Shifted Gaussian Pulses**:

$$V(t) = \underbrace{\beta \cdot t}_{\text{Global Rise}} + \sum_{i=1}^{N} \underbrace{\alpha_i \cdot \exp\left(-\frac{(t-\tau_i)^2}{2\sigma^2}\right)}_{\text{Local Crisis } i} \tag{3}$$

This composite function operationalises the following narrative dynamics:

- **Global Slope (β):** Ensures that the tension baseline rises continuously. Even after a crisis is resolved, the narrative never returns to the initial state of safety.
- **Event Amplitude (α_i):** The height of the i-th crisis. To strictly enforce the rule of "Rising Action," we apply the inequality constraint $\alpha_{i+1} > \alpha_i$. In plain terms, this mathematical inequality ensures that the next problem is always bigger than the last one, preventing the story from losing momentum.
- **Temporal Shift (τ_i):** The specific moment in time when Crisis i occurs.

Algorithmic Derivative Check. The generative agent monitors the derivative $V'(t)$. A positive spike in $V'(t)$ signals the onset of a new crisis routine, while the subsequent negative slope signals the "Falling Action" of that specific micro-narrative.

2.3 Narrative Structure: In Medias Res

2.3.1 History and Theoretical Framework

In Medias Res (Latin for "into the midst of things") is a technique dating back to Homer's *The Odyssey* [7]. The narrative initialises at a point of high tension (a Middle Crisis), omitting

the setup. As shown in Figure 3, the context is subsequently provided via non-linear temporal shifts (flashbacks).

1. **Middle Crisis:** The simulation begins at State S_k (High Tension).
2. **Rising Action:** The narrative proceeds forward in time.
3. **Exposition/Flashback:** Concurrent backward motion is required to explain the causality of S_k.
4. **Climax:** The temporal threads converge.

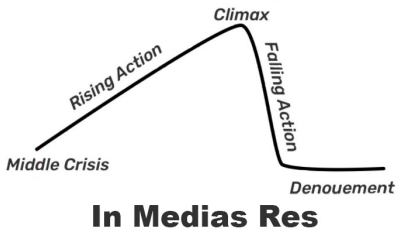

Figure 3: In Medias Res: The narrative starts at a local maxima (S_k), requiring bi-directional planning to resolve both future consequences and past causality.

By alternating between these queues (Algorithm 3), the AI effectively "unzips" the timeline from the centre outward, ensuring the Flashback (generated later) perfectly explains the Opening Scene (generated first).

Algorithm 3: InMediasRes Bi-Directional Generation

Input: StartNode S_k (The Crisis)
ForwardQueue ← $[S_k]$;
BackwardQueue ← $[S_k]$;
`// Forward Pass: Standard Prediction`
while *State* ≠ *Resolution* **do**
  NextState ← Predict(*CurrentState*);
  ForwardQueue.push(*NextState*);
end
`// Backward Pass: Abductive Inference`
while *State* ≠ *Origin* **do**
  Preconditions ← GetPreconditions(*CurrentState*);
  `// Find past event via Bayes Maximisation`
  PriorState ← SolvePosterior($P(H|O)$);
  BackwardQueue.push(*PriorState*);
end

2.3.2 Algorithmic Implementation

This structure presents a fundamental architectural challenge. Standard Large Language Models (LLMs) are **Autoregressive**, meaning they operate with a strictly forward-facing temporal bias (consuming t to predict $t+1$). This creates a computational asymmetry: while they can simulate consequences, they cannot natively infer causes. To implement *In Medias Res*, where the story starts at the Climax ($t = 50$) and expands outward in both directions, we implement a **Bi-Directional Planner**.

The system initialises the narrative state S_{start} at the peak of a local crisis (e.g., "The bomb timer reads 00:05"). It then manages two divergent processing queues:

1. The Forward Queue (Deductive Reasoning). This functions as a standard simulator. Given the crisis S_{start}, what happens next?

- *Logic:* Cause → Effect.
- *Operation:* The AI predicts the immediate consequences (e.g., The hero cuts the red wire).

2. The Backward Queue (Abductive Reasoning). This process functions analogously to a detective reconstructing a crime scene. The detective observes the final state (the body) and must deduce the sequence of events that led to it. It relies on **Abductive Inference**: finding the most plausible explanation for an observed phenomenon.

- *Logic:* Effect → Cause.
- *Operation:* The AI must generate a valid antecedent state S_{-1} that logically results in S_{start}.
- *Example:* If the start state is "Hero is hanging from a cliff," the Backward Queue generates the hypothesis "Hero was chased by wolves" rather than "Hero was sleeping in bed," because the latter does not logically entail the cliff state.

2.3.3 Mathematical Modelling Proposition: Bayesian Abduction and Inverse Probability

Standard generative models operate forward in time ($P(Future|Past)$). However, *In Medias Res* requires **Inverse Causal Reasoning**: we are given the effect (the Opening Scene) and must infer the cause (the Backstory).

We operationalise this using Bayes' Theorem to solve for the Posterior Probability $P(H|O)$, where H is a candidate History sequence and O is the fixed Opening state:

$$\text{maximise } P(H|O) = \frac{\overbrace{P(O|H)}^{\text{Causal Consistency}} \cdot \overbrace{P(H)}^{\text{Narrative Prior}}}{P(O)} \tag{4}$$

This formula decomposes the generation task into two distinct evaluation metrics:

1. The Likelihood Function P(O|H) (Causal Consistency). This term answers the question: *If this backstory occurred, does the opening scene logically follow?* This acts as a strict logical gate. For example, let O contain the predicate `Status(Hero, Wounded)`.

- **Scenario A:** The AI generates a history H_A where the hero slept peacefully all night. Here, $P(O|H_A) \approx 0$ because the history fails to explain the wound.
- **Scenario B:** The AI generates a history H_B where the hero fought a duel. Here, $P(O|H_B) \approx 1$.

In our architecture, this is calculated via a **Natural Language Inference (NLI)** module. If H does not logically entail O, the candidate history is rejected as a "plot hole."

2. The Prior P(H) (Narrative Plausibility). This term answers the question: *Is this backstory coherent on its own, independent of the opening?* Even if a history explains the opening, it must adhere to the internal logic of the world.

- If H involves a *Deus Ex Machina* (e.g., "A wizard suddenly appeared and teleported the hero."), the model penalises this as a low-probability event within a realistic genre setting.

By maximising the product of these two terms, the system converges on a backstory that is both logically consistent with the present crisis (O) and narratively sound (H).

2.4 Narrative Structure: The Three-Act Structure

2.4.1 History and Theoretical Framework

The Three-Act Structure, codified by Syd Field [8], represents the dominant paradigm in cinematic storytelling. It partitions the narrative into three distinct units separated by "Plot Points" (see Figure 4). These points function as irreversible thresholds; traversing them alters the narrative state permanently [9].

1. **Act I (Setup):** Concludes with **Plot Point 1**, where the protagonist commits to the journey.
2. **Act II (Confrontation):** The central conflict phase. Includes the **Midpoint** (a shift in protagonist agency) and concludes with **Plot Point 2** (the lowest point of defeat).
3. **Act III (Resolution):** The final confrontation and conclusion.

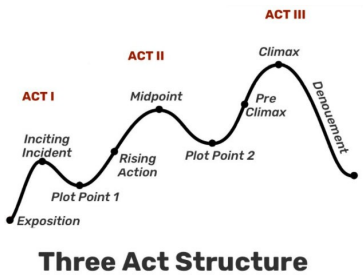

Figure 4: Three-Act Structure: A partitioned graph where Plot Points act as directed edges preventing regression to previous states.

2.4.2 Algorithmic Implementation

We model the narrative space as a **Directed Acyclic Graph (DAG)**. In a standard state machine, transitions are often reversible (e.g., a character can walk from the *Kitchen* to the *Hallway* and back). However, the Three-Act structure relies on the concept of the "Point of No Return." To enforce this computationally, we treat the narrative not as a static map, but as a **Dynamic Topology** that evolves as the agent traverses it.

The Pruning Operation ("Burning the Ships"). The transition between acts is not merely a movement but a structural alteration. We can conceptualise this through the military strategy of "burning the bridges" after crossing them. Once the protagonist leaves the "Ordinary World" of Act I, the path backward must be destroyed to force the narrative forward.

Technically, we partition the total graph G into three subgraphs ($G_{\text{Setup}}, G_{\text{Conflict}}, G_{\text{Resolution}}$) connected strictly by singular **Bridge Edges** (P_1, P_2). As the generative agent traverses the bridge P_1:

1. **Detection:** The system identifies that the current state $S_t \in G_{\text{Conflict}}$.
2. **Deletion:** The system identifies all edges e_{ji} where the source is in the current Act and the target is in the previous Act.
3. **Topological Collapse:** These edges are deleted from the adjacency matrix ($A_{ji} \leftarrow 0$).

This operation prevents the AI from "getting homesick." Even if the probabilistic model generates a high likelihood for a token like "Go home" (because "home" is a safe, high-probability concept), the planner rejects it because the edge to "Home" no longer exists in the graph topology. This forces the narrative vector v_{story} to move exclusively forward towards the climax (Algorithm 4).

Algorithm 4: Three-Act Directed Generation

Input: Graph G, StartNode S_{start}, Thresholds P_1, P_2

$CurrentAct \leftarrow 1$;
$Node \leftarrow S_{\text{start}}$;
while $CurrentAct \leq 3$ **do**
  $Node \leftarrow G$.Traversal($Node$);
  if $Node == P_1$ ***and*** $CurrentAct == 1$ **then**
    $CurrentAct \leftarrow 2$;
    `// Cut the connection to the start`
    G.PruneEdges(Source = Act2, Target = Act1);
  end
  if $Node == P_2$ ***and*** $CurrentAct == 2$ **then**
    $CurrentAct \leftarrow 3$;
    `// Cut the connection to the middle`
    G.PruneEdges(Source = Act3, Target = Act2);
  end
  Render($Node$);
end

2.4.3 Mathematical Modelling Proposition: Subgraph Connectivity and Conflict Density

To distinguish the pacing of the three acts, we analyse the topological properties of the narrative graph. We define a subset of vertices $V_{\text{conflict}} \subset V$ representing states of high antagonism (e.g., "Hero Trapped," "Item Lost"). We then define **Conflict Density** (ρ) as the probability that a randomly selected node traversal leads to a conflict state:

$$\rho(G_k) = \frac{|\{(u, v) \in E_k \ : \ v \in V_{\text{conflict}}\}|}{|E_k|} \tag{5}$$

This metric imposes the structural inequality $\rho(G_2) > \rho(G_3) > \rho(G_1)$.

- **Act 1 (Low Density):** The narrative focuses on exposition; edges connect to neutral information states.
- **Act 2 (High Density):** Known as the "Confrontation," this subgraph has a high friction coefficient. Almost every edge should carry a risk of transitioning to a negative state, mathematically simulating the "Road of Trials."

- **Act 3 (Medium Density):** While intensity is high, complexity collapses as the graph trajectory straightens towards the singularity of the Climax.

Generative Sampling Bias. In our architecture, $\rho(G_k)$ functions as a **Logit Bias** parameter. When the planner detects the agent is in G_2, it artificially boosts the logits for adversative tokens ("But," "However," "Suddenly"). This forces the neural network to hallucinate obstacles at a higher frequency, preventing the story from resolving too easily.

2.5 Narrative Structure: The Seven-Point System

2.5.1 History and Theoretical Framework

Dan Wells' Seven-Point System [10] is a structural framework often utilised for outlining due to its focus on dramatic state changes. A key characteristic is that it is often planned in reverse, beginning with the Resolution. Figure 5 illustrates these high-contrast transitions.

1. **Hook:** The starting state (often the antithesis of the resolution).
2. **Plot Point 1:** The call to action.
3. **Pinch Point 1:** Introduction of the antagonist force.
4. **Midpoint:** The transition from reactive to active agency.
5. **Pinch Point 2:** The "All is Lost" moment.
6. **Plot Point 2:** Acquisition of the solution.
7. **Resolution:** Conclusion of the conflict.

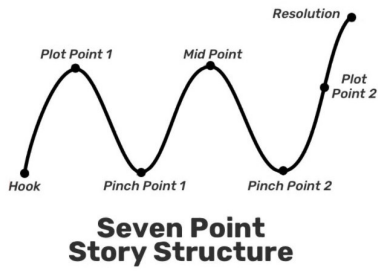

Figure 5: The Seven-Point System: A sparse-reward environment where significant narrative value is only achieved at specific structural milestones.

2.5.2 Algorithmic Implementation

The Seven-Point structure is ideally mapped to a **Reinforcement Learning (RL)** environment to enforce **long-horizon planning**. Traditional sequence models (LLMs) are optimised for local reward (predicting the next correct word), leading to the "horizon problem" where global narrative goals are lost.

The Problem of Dense Reward. In open-ended generation, using a "dense" reward (e.g., scoring every generated sentence based on readability or sentiment) is prone to **reward hacking**. The agent learns to exploit flaws in the scoring function by generating repetitive, semantically safe, or simple text sequences to maximise its immediate score, resulting in a predictable and dull narrative.

The Sparse Reward Solution. We solve this by defining the structural moments (the Seven Points) as **Sparse Rewards**. To understand the utility of this approach, consider the analogy of navigating a dense forest. A standard Language Model operates like a hiker looking only at their feet, choosing the easiest next step without knowledge of the destination. In contrast, an RL agent operates with a compass, ignoring the immediate ease of the path to focus on the distant mountain peak (the Plot Point).

1. **Transition Steps ($r = -0.1$):** For every scene generated between the milestones, the agent receives a slight negative reward (-0.1). This penalty discourages the agent from rambling or wasting narrative time.
2. **Milestone Steps ($r = +100$):** A high-magnitude positive reward ($+100$) is only issued when the current narrative state vector successfully achieves the semantic and logical conditions of the target milestone (e.g., the "Midpoint").

This forces the RL agent to engage in strategic search across the entire narrative state space. The agent learns that the only path to maximising its total reward is by proactively generating a coherent sequence of events that successfully leads to the distant, high-value **Attractor States** (the Milestones), rather than seeking short-term local gains (Algorithm 5).

Algorithm 5: Seven-Point Reinforcement Generation

Input: StartNode S_0, Milestones M

$State \leftarrow S_0$;
foreach $Target \in M$ **do**
  while $State \neq Target$ **do**
    $Action \leftarrow \pi(State)$ // AI guesses next word
    $State \leftarrow$ Transition($State, Action$);
    $Reward \leftarrow -0.1$ // Small penalty for taking too long
  end
  $Reward \leftarrow +100$ // Huge bonus for hitting the Plot Point
  // Optimise Policy via Bellman (Eq. 7)
  UpdateWeights($Reward$);
end

2.5.3 Mathematical Modelling Proposition: Markov Decision Processes and Value Landscapes

We formalise the narrative environment as a Markov Decision Process (MDP), defined by the tuple $\mathcal{M} = \langle S, A, T, R, \gamma \rangle$.

- S: The continuous semantic state of the narrative (represented by the Transformer's hidden state vectors).
- A: The set of possible narrative beats or generated sentences.
- $T(s'|s, a)$: The transition probability (the language model's inherent probability distribution).
- $R(s)$: The sparse reward function (defined by the Seven Point structure).
- γ: The discount factor, determining how far ahead the agent plans.

The Value Function as a Narrative Compass. The core challenge in sparse-reward environments is the "Credit Assignment Problem": *how does the agent know if a sentence written on Page 10 is good if the reward (the Climax) isn't until Page 100?*

We solve this using the Bellman Equation to learn a **Value Function** $V^{\pi}(s)$. This function estimates the expected cumulative reward from the current state s:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} R_{t+k+1} \mid S_t = s\right] \tag{6}$$

Visually, this creates a **Value Landscape** or topographic map of the story. In this topological model, the Seven Points (e.g., Midpoint, Pinch Point) act as **Global Maxima** (high peaks) in the value landscape, while irrelevant tangential subplots act as valleys (low value).

Gradient Ascent towards the Plot Point. The generative agent operates by performing **Gradient Ascent** on this landscape. When selecting the next narrative action a, the agent does not merely select the most probable token (as a standard LLM does); it selects the action that maximises the transition to a higher-value state:

$$a_{\text{optimal}} = \arg\max_{a}\left(R(s,a) + \gamma \sum_{s'} T(s'|s,a)V(s')\right) \tag{7}$$

This creates a "gravitational pull." Even if the agent is far from a Pinch Point, the gradient $\nabla V(s)$ points the trajectory in the direction that minimises the geodesic distance to the next structural milestone, effectively solving the "wandering plot" problem.

2.6 Narrative Structure: The Hero's Journey

2.6.1 History and Theoretical Framework

The Monomyth, identified by Joseph Campbell [11] and adapted for screenwriting by Christopher Vogler [12], is a universal template observed in mythology and modern media. Campbell argued that this structure is not merely a collection of tropes, but a psychological map of human maturation. It represents the universal human experience of leaving the comfort of the known world, facing the trials of the unknown, and returning transformed.

The structure is defined by a cycle of 12 distinct stages (see Figure 6), categorised into three phases: Separation, Initiation, and Return.

1. **Ordinary World:** Status quo and limited awareness.
2. **Call to Adventure:** Disruption of comfort.
3. **Refusal of the Call:** Fear of change.
4. **Meeting the Mentor:** Acquiring guidance or supplies.
5. **Crossing the Threshold:** Commitment to the journey; the point of no return.
6. **Tests, Allies, Enemies:** Exploring the new world.
7. **Approach to Inmost Cave:** Preparing for the danger.
8. **The Ordeal:** Death and rebirth; the central crisis.
9. **Reward:** Seizing the objective (the sword, the elixir, the knowledge, etc.).
10. **The Road Back:** Urgency of escape or return.
11. **Resurrection:** Final exam; a high-stakes climax where the hero proves they have changed.
12. **Return with the Elixir:** Reintegration into society with new wisdom.

Figure 6: The Hero's Journey: A cyclical finite state machine with 12 distinct states requiring inventory prerequisites for transition.

2.6.2 Algorithmic Implementation

The Hero's Journey is modelled as a **Gated Finite State Machine (Gated FSM)**, a concept utilised in interactive narrative systems [13]. This is a more robust version of a standard FSM designed to handle complex narrative dependencies. A traditional FSM determines the next state solely based on the current state and the input signal. However, the Monomyth requires **Stateful Memory** to track the hero's growth.

The Gated Mechanism. To understand the necessity of this "Gated" architecture, consider the progression logic of a Role-Playing Game (RPG). In such systems, a player often encounters a locked door (a narrative threshold) that cannot be opened until a specific key (a narrative prerequisite) is found in a dungeon. The door is visible, and the path exists, but the transition is blocked until the inventory condition is met.

We define the system with a global memory component called the **Inventory Vector** ($\mathbf{I}_t$), which stores all logical facts, items, and character traits acquired up to time t. The narrative flow is then controlled by logical gates on the transition edges:

1. **Precondition Matrix ($\mathscr{R}$):** Every transition edge T_{ij} has a specific set of required elements, $\mathscr{R}_{ij}$, derived from the Monomyth structure (e.g., leaving the Ordinary World requires accepting the Call).
2. **Causality Check:** Before the generator can proceed, the Symbolic Planner must verify that the current Inventory Vector $\mathbf{I}_t$ satisfies $\mathscr{R}_{ij}$.
3. **Transition Lock:** If $\mathbf{I}_t$ does not contain the required prerequisite, the transition edge T_{ij} is temporarily masked (set to zero probability), effectively locking that stage of the journey.

Rerouting and Sub-Quest Generation. If the narrative flow stalls because a prerequisite is missing (e.g., the system tries to generate "Meeting the Mentor" but the men-

tor is logically absent), the agent is temporarily rerouted into a sub-quest generation loop. This sub-loop is constrained to generate only scenes that result in the acquisition of the missing precondition. Once the inventory is updated (`Inventory.Add(State.Reward)`), the transition gate unlocks, and the main Monomyth progression resumes (Algorithm 6). This prevents narrative sequence breaking (e.g., proceeding to the "Ordeal" without first acquiring the necessary knowledge from the "Mentor").

Algorithm 6: Monomyth Gated Generation

Input: Graph G, Inventory I_0, StartNode S_0

```
State ← S_0;
Inventory ← I_0;
while State ≠ ReturnWithElixir do
    PossibleMoves ← G.GetAdjacency(State);
    for Move ∈ PossibleMoves do
        // Check if we have the required items to
           proceed
        if Inventory ⊇ Move.Preconditions then
            State ← Move.Target;
            Inventory.Add(State.Reward);
            break;
        end
    end
end
```

2.6.3 Mathematical Modelling Proposition: Gated State Transition Matrices

To enforce the strict sequentiality and causal logic of the Monomyth, we model the narrative not as a simple Markov Chain, but as a **Gated Finite State Machine**.

We define the connectivity of the 12 stages using a base Adjacency Matrix $A \in \{0,1\}^{12\times 12}$, where $A_{ij} = 1$ implies a path exists from Stage i to Stage j. However, mere connectivity is insufficient; the hero must be *qualified* to traverse the edge.

We introduce a **State Vector** $\mathbf{x}_t$ (representing the current narrative stage) and an **Inventory/Fact Vector** $\mathbf{I}_t$ (representing accumulated knowledge, items, or emotional states).

The effective Transition Matrix T is computed dynamically at each time step by applying a **Causality Mask** M:

$$T_{ij}(t) = A_{ij} \cdot \underbrace{\mathbb{I}(\mathbf{I}_t \supseteq \mathscr{R}_{ij})}_{\text{Causality Mask}} \tag{8}$$

Where:

- $\mathscr{R}_{ij}$: The set of prerequisites required to move from node i to node j (e.g., "Must have met Mentor," "Must have Sword").
- $\mathbb{I}(\cdot)$: The Indicator Function, which returns 1 if the condition is true and 0 otherwise.
- $\mathbf{I}_t \supseteq \mathscr{R}_{ij}$: A subset check confirming the narrative memory contains all necessary preconditions.

Prevention of Logical Hallucination. This multiplication operation may appear abstract, but it functions simply as a **logical checklist**. The term $\mathbb{I}(\cdot)$ asks: "Does the hero have the key?" If the answer is no (0), the entire probability for that path becomes zero, regardless of how close the hero is geographically. Even if the neural network predicts that the token "Resurrection" (Stage 11) is statistically likely after "Approach the Cave" (Stage 7) due to semantic similarity, the matrix enforces a hard zero probability:

$$P(\text{Stage}_{11}|\text{Stage}_7) = 0 \quad \text{because} \quad \text{OrdealComplete} \notin \mathbf{I}_t \tag{9}$$

This mathematically guarantees **Causal Consistency**, rendering it impossible for the AI to generate an ending before the conflict has been resolved.

2.7 Narrative Structure: Save the Cat

2.7.1 History and Theoretical Framework

Derived from commercial screenwriting, Blake Snyder's *Save the Cat!* [14] represents the most rigid of these structures. It outlines 15 specific "beats" (narrative moments) mapped to precise page numbers in a standard 110-page screenplay. As illustrated in Figure 7, this structure imposes several unique constraints beyond simple plot points:

1. **The Pacing Curve:** The narrative must adhere to specific temporal milestones (e.g., the "Midpoint" must occur exactly at 50% of the length).
2. **The B-Story (Red Arrow):** Snyder emphasises a secondary plot thread—often a love story or a relationship arc—that begins at the start of Act II. This thread carries the moral theme of the story, running parallel to the main action (A-Story) before intersecting with it at the climax.
3. **Thematic Rhyming (Green Arrows):** The structure demands a specific symmetry between the "Opening Image" and the "Final Image." These two moments must be "mirror opposites," visually demonstrating the transformation the protagonist has undergone (e.g., a hero starts alone in a dark room and ends surrounded by friends in a bright room).

Figure 7: Save the Cat: An A* Search landscape. The red arrow indicates the parallel B-Story thread, while green arrows indicate the required semantic symmetry (rhyming) between the start and end states.

Moreover, each beat is assigned a strict page range (e.g., the "Catalyst" must occur on Page 12). This transforms the narrative generation task into a **Constrained Pacing Problem**, where the AI must not only generate coherent text but also ensure that specific plot events occur at predetermined intervals. The 15 beats of the *Save the Cat* structure are as follows:

1. **Opening Image (1):** The visual "before" snapshot of the protagonist's world.
2. **Theme Stated (5):** A secondary character poses a question or statement that is the thematic argument of the story.
3. **Set-Up (1-10):** Exposition of the hero's flaws and the status quo.
4. **Catalyst (12):** The inciting incident that disrupts the status quo.
5. **Debate (12-25):** The hero resists the call to action; a period of doubt.
6. **Break into Two (25):** The hero makes a proactive choice to enter the "Special World" (Act II).
7. **B-Story (30):** Introduction of the relationship character (love interest/mentor) who carries the theme.
8. **Fun and Games (30-55):** The "promise of the premise" where the hero explores the new world (often the source of trailer moments).
9. **Midpoint (55):** A major plot twist raises the stakes; often a "false victory" or "false defeat."
10. **Bad Guys Close In (55-75):** Internal and external forces tighten their grip; the hero's plan begins to fail.
11. **All is Lost (75):** The absolute low point; often involves a symbolic or literal death.
12. **Dark Night of the Soul (75-85):** The hero wallows in their defeat before realising the thematic truth.
13. **Break into Three (85):** The "Aha!" moment where the true solution is discovered.
14. **Finale (85-110):** The final confrontation where the hero executes the new plan and proves their transformation.
15. **Final Image (110):** The visual "after" snapshot, structurally mirroring the Opening Image to show change.

2.7.2 Algorithmic Implementation

The rigid page counts of the *Save the Cat* methodology convert story generation from a simple sequence prediction problem into a constrained **Optimal Path Search**. We utilise the **A*** (**A − Star**) algorithm, an informed search technique widely used in pathfinding [5].

An Analogy. To understand the necessity of A* here, consider the difference between wandering and navigating. A standard Language Model functions like a driver wandering a city based on which street looks interesting (local probability). A *Save the Cat* planner functions like a **GPS Navigation System** with a strict constraint: "You must arrive at the destination in exactly 10 minutes." If the GPS detects traffic (a narrative tangent) that will delay the arrival, it forces a route change to get back on schedule. The visual curve in Figure 7 represents the optimal highway; the algorithm's job is to pull the wandering driver back onto this line.

The Search Space and Heuristic. The narrative space is a complex, combinatorial search graph where each potential sentence or scene represents a potential transition edge. Snyder's 15 beats function as the necessary intermediate goal nodes. The core of the A* efficiency lies in its **Heuristic Function** $h(n)$, which estimates the remaining distance to the goal. This heuristic is essential for pruning unproductive narrative branches.

Priority Queue Management. The algorithm maintains a **Priority Queue** of narrative segments (nodes) that could potentially lead to the next beat. The priority of a node is determined by its total estimated cost $f(n) = g(n) + h(n)$.

1. $g(n)$ (Actual Cost): The accumulated word/token count of the path generated so far.
2. $h(n)$ (Heuristic Cost): The estimated penalty for deviation from the planned pace.

Pacing Corridor Enforcement. The generative agent strictly adheres to a **Pacing Corridor**. For every beat B_i (with a target token count T_i), the heuristic $h(n)$ acts as a "gravitational pull" or rubber band:

- **High Semantic Distance:** If the text is nearing the target count T_i but has not semantically triggered the plot point (e.g., the hero hasn't actually *failed* at the "All is Lost" beat), the heuristic cost spikes, prioritising actions that force the narrative event to happen immediately.
- **Temporal Overrun:** If the token count exceeds T_i without triggering the beat, the cost $f(n)$ rapidly approaches infinity. This ensures that the branch is immediately deprioritised by the queue and effectively pruned, forcing the system to backtrack and regenerate a shorter, more impactful segment that hits the beat precisely on time (Algorithm 7).

This mechanism guarantees the final output respects the critical temporal synchronisation demanded by commercial screenwriting.

Algorithm 7: SaveTheCat Heuristic Generation

Input: BeatSheet B, TargetCounts T, StartNode S_0

$FullStory \leftarrow \emptyset$;
$CurrentState \leftarrow S_0$;
// Iterate through transitions between the 15 beats
for $i \leftarrow 0$ **to** 13 **do**
    $Goal \leftarrow B[i+1]$;
    $TargetLength \leftarrow T[i+1]$;
    // Constraint: Hit Goal exactly at TargetLength
    $Path \leftarrow$ AStarSearch($CurrentState$, $Goal$, $TargetLength$);
    $FullStory$.append($Path$);
    $CurrentState \leftarrow Path$.EndNode;
end

2.7.3 Mathematical Modelling Proposition: Heuristic Search and Pacing Constraints

To enforce the rigid pacing of the *Save the Cat* structure (e.g., the "All Is Lost" moment must occur exactly at 75% of the total length), we frame story generation as a pathfinding problem on a graph. We utilise the A* Search algorithm, which selects the optimal next narrative segment n by minimising the total

estimated cost $f(n)$:

$$f(n) = \underbrace{g(n)}_{\text{Past Cost}} + \underbrace{h(n)}_{\text{Future Heuristic}} \tag{10}$$

We redefine these standard terms for the domain of Computational Narratology:

1. The Accumulated Cost $g(n)$. This represents the Temporal Expenditure of the narrative so far. In a Large Language Model, we measure this in tokens.

$$g(n) = \frac{\text{CurrentTokenCount}(n)}{\text{TotalTargetTokens}} \tag{11}$$

As the story progresses, $g(n)$ increases linearly. This prevents the model from looping infinitely; every word "costs" something, consuming the limited budget of the story's length.

2. The Heuristic $h(n)$. This is the critical innovation. The heuristic estimates the cost to reach the next Plot Point (Goal). It is a composite of two distinct distance metrics:

$$h(n) = \alpha \cdot \underbrace{D_{\text{sem}}(n, Goal)}_{\text{Semantic Distance}} + \beta \cdot \underbrace{P_{\text{pace}}(n, Goal)}_{\text{Pacing Penalty}} \tag{12}$$

- **Semantic Distance (D_{sem}):** calculated as the cosine distance between the current state vector and the target archetype vector (e.g., *how far is the current text from the concept of "Defeat"?*).
- **Pacing Penalty (P_{pace}):** A non-linear penalty function.

The "Infinite Wall" Pruning Mechanism. To enforce the 75% mark, we define P_{pace} as an exponential wall. Let T_{target} be the target word count for the "All is Lost" beat.

- **Under-shoot:** If the AI tries to trigger the beat too early (e.g., at 50%), the heuristic adds a penalty, forcing the AI to generate "filler" or sub-plots to pad the time.
- **Over-shoot:** If the current length exceeds T_{target} and the beat has not been triggered, $h(n) \to \infty$.

$$\begin{aligned} &\textbf{If } \text{Count}(n) > T_{\text{target}} \\ &\quad \textbf{AND } \text{Beat} \neq \text{Triggered} \\ &\implies h(n) = \infty \end{aligned} \tag{13}$$

This effectively prunes the branch. The search algorithm abandons this narrative thread as a "dead end" and backtracks to a previous state to attempt a more concise path that lands the emotional climax exactly on the required page.

3 Archetypes as Latent Priors: Vector Quantisation of Character

Having established the algorithmic flow of narrative events (the "Plot") in Section 2, we must now define the entities that navigate these graphs: the characters. In traditional literary criticism, characters are typically regarded as unique, ineffable artistic creations. However, to construct a generative artificial intelligence capable of storytelling, we cannot treat every character as a singularity.

For the purposes of Computational Narratology, we posit that characters are probabilistic instantiations of stable, high-level categories known as **Archetypes**. To model this, we map the psychological theories of Carl Jung to the computer science concepts of **Vector Space Models (VSM)** and **Latent Space Clustering**. We argue that what a psychologist identifies as an "Archetype" is computationally isomorphic—identical in structure—to what a data scientist identifies as a *Centroid* in a high-dimensional feature space.

3.1 Jungian Psychology: The Structure of the Psyche

3.1.1 Theoretical Framework

Swiss psychiatrist Carl Jung challenged the empiricist notion that the human mind is a *tabula rasa* (blank slate). In *The Structure and Dynamics of the Psyche* [15], he proposed that the psyche possesses an inherited, biologically determined structure known as the **Collective Unconscious**. This is not a repository of specific memories, but rather a system of "primordial images" or "inherited possibilities of representation."

To explicate this concept, Jung utilised a crystallographic analogy: the archetype is not the concrete crystal itself, but the invisible lattice structure that dictates how the crystal must form [16]. In the context of AI, archetypes function not as content, but as **Priors** or **Weight Initialisations**. They serve as the pre-existing constraints that shape the behavioural learning process of a social agent.

3.1.2 Structural vs. Functional Archetypes

It is critical to distinguish between the *components* of the psyche and the *roles* assumed by the psyche. Jung initially identified structural elements—functional components of the mind's operating system.

- **The Persona:** The social mask. This represents the curated interface the individual presents to society to ensure acceptance.
- **The Shadow:** The repository of repressed or rejected traits. If a protagonist's conscious vector represents bravery, the Shadow represents the inverse vector (cowardice). Computationally, this is defined as:

$$v_{\text{shadow}} \approx -1 \cdot v_{\text{ego}} \tag{14}$$

- **The Anima/Animus:** The anthropomorphic representation of the subconscious. It serves as a bridge to the collective unconscious, often projected onto a romantic partner.
- **The Self:** The centralisation of the total psyche. It represents the objective of the "Individuation Process," which corresponds to the Resolution state in narrative generation.

While the structural elements define the internal machinery, the specific character "roles" we see in stories are manifestations of these drives. Although the potential number of archetypes is theoretically infinite as they represent the myriad experiences of the species [17], post-Jungian scholars have codified them into a manageable taxonomy. Most notably,

Pearson [18] derived **The Twelve Archetypes** system, which categorises character motivations into three sets:

1. **The Ego Set (Preparation):** Characters driven by social integration.
   - *The Innocent:* Desires safety; fears abandonment.
   - *The Orphan:* Desires belonging; fears exclusion.
   - *The Warrior:* Desires mastery; fears weakness.
   - *The Caregiver:* Desires to help others; fears selfishness.
2. **The Soul Set (Journey):** Characters driven by individual development.
   - *The Explorer:* Desires freedom; fears entrapment.
   - *The Rebel:* Desires revolution; fears powerlessness.
   - *The Lover:* Desires intimacy; fears isolation.
   - *The Creator:* Desires innovation; fears mediocrity.
3. **The Self Set (Return):** Characters driven by order and wisdom.
   - *The Jester:* Desires enjoyment; fears boredom.
   - *The Sage:* Desires truth; fears deception.
   - *The Magician:* Desires transformation; fears unintended consequences.
   - *The Ruler:* Desires control; fears chaos.

3.2 Computational Evidence: Latent Personas

3.2.1 Statistical Validation

For decades, these classifications were regarded as unscientific. However, modern Natural Language Processing (NLP) has provided empirical evidence supporting Jung's hypothesis. The seminal validation was provided by Bamman, O'Connor, and Smith (2013) in their paper *Learning Latent Personas of Film Characters* [19].

The authors applied **unsupervised Bayesian modelling** (specifically Dirichlet Process Mixtures) to a dataset of 42,000 movie character summaries. The term "unsupervised" implies that the algorithm was provided with no prior knowledge of Jungian theory or literary tropes; it was tasked with identifying patterns solely based on the lexical dependencies of character actions.

The model discovered K "Latent Personas" (clusters) that aligned remarkably with standard literary archetypes:

- **Cluster 1 (The Protector/Father):** Defined by dependency paths involving verbs such as *save, protect, father.*
- **Cluster 2 (The Femme Fatale):** Defined by verbs such as *seduce, kill, betray.*
- **Cluster 3 (The Sidekick):** Defined by verbs such as *follow, help, run.*

This suggests that archetypes are not merely cultural inventions, but **Statistical Attractors**. In the high-dimensional vector space of human behaviour, narratives naturally gravitate towards these specific clusters.

3.3 Mathematical Modelling: Archetypal Centroids

To operationalise this for a generative AI agent, we must formally define character creation as a **Vector Quantisation** problem.

First, we establish a **Vector Space**. As demonstrated by Mikolov et al. [20] (the creators of Word2Vec), semantic concepts can be represented as vectors in a space $\mathbb{R}^d$. To render this intuitive, one may visualise this high-dimensional vector space as a **Map of Personality**. In this spatial model, characters are not isolated points but residents of specific semantic "neighbourhoods." For instance, all heroic characters (e.g., Harry Potter, Luke Skywalker) cluster in one region, while villainous characters (e.g., Voldemort, Darth Vader) cluster in a distant, opposing region. In this geometric space, semantic relationships are linear. The classic example is the arithmetic operation: $Vector(King) - Vector(Man) + Vector(Woman) \approx Vector(Queen)$.

We define a character C as a vector v_c, which represents the aggregate average of their attributes, dialogue, and actions. We subsequently define the set of Archetypes $\mathscr{A} = \{A_1, A_2, ..., A_k\}$ as the **Centroids** (the geometric centres) of the clusters identified in the dataset of successful narratives.

$$\mu_k = \frac{1}{|C_k|} \sum_{v_c \in C_k} v_c \tag{15}$$

Here, μ_k is the centroid vector for Archetype k (e.g., The Innocent). In our analogy, this centroid acts as the "Town Square" of the neighbourhood—the purest mathematical representation of that archetype.

3.3.1 Generative Constraint

When an AI generates a character, it often exhibits "drift"—inconsistent behaviour where a character functions as a villain in the initial state and a mentor in a subsequent state without narrative justification. To mitigate this, we impose a **Distance Constraint**. We force the generative model to minimise the Euclidean distance between the generated character's current behaviour vector (v_{gen}) and their assigned Archetypal Centroid (μ_{target}).

$$\text{Loss}_{\text{character}} = \|\mathbf{v}_{\text{gen}} - \mu_{\text{target}}\|^2 \tag{16}$$

This function operates as a form of **Magnetic Attraction**. By incorporating this term into the Neural Network's loss function, we mathematically penalise the AI whenever the character attempts to leave their assigned neighbourhood. If a character designated as a "Warrior" begins to act like a "Coward" (moving away from the centroid), the error increases, effectively pulling the narrative trajectory back towards the archetypal centre to ensure behavioural consistency.

4 The Cognitive Substrate: Narrative as Active Inference

We have defined the Algorithms via Section 2 and the Data Structures via Section 3. We must now address the Hardware: the human brain. *Why do these specific structures and*

archetypes persist across millennia? We argue that they are evolutionary optimisations for the brain's predictive architecture. We frame this using the **Free Energy Principle (FEP)** and **Predictive Processing (PP)**.

4.1 The Brain as a Hierarchical Prediction Machine

4.1.1 Theoretical Framework

Cognitive scientist Andy Clark describes the brain not as a passive receiver of sensory input, but as an active "Prediction Machine" [21]. The brain maintains a hierarchical generative model of the world and is continuously generating top-down predictions to explain bottom-up sensory data.

According to Karl Friston's Free Energy Principle [1], the biological imperative of any self-organising system is to minimise **Free Energy**. In this information-theoretic context, Free Energy is effectively synonymous with "Surprisal" or **Entropy**.

$$F \approx -\ln P(o|m) \tag{17}$$

Where o represents the observation (sensory reality) and m represents the internal model (expectation). If the brain predicts incorrectly, F is high (resulting in anxiety or surprise). If the brain predicts correctly, F is low.

To understand this intuitively, one can regard the brain as an organ driven by a strict energy budget. Processing unexpected stimuli—surprisal—requires a significant spike in neural activity and metabolic consumption. Therefore, the brain is biologically incentivized to be "lazy" or, more accurately, **metabolically efficient**. It hates surprises because surprises are expensive. It seeks to predict the world accurately so it can remain in a low-energy state.

4.2 Narrative as Data Compression

Raw reality is stochastic, chaotic, and possesses high entropy. It is computationally expensive for a biological brain to memorise or predict a random sequence of events.

- **Real World:** $S = \{e_1, e_2, ..., e_n\}$. The probability of event e_{t+1} occurring given e_t is low.
- **Narrative:** $S' = \text{Structure}(S)$. The probability of e_{t+1} occurring given e_t is high. (e.g., If the hero enters the cave, the probability of encountering the dragon is near 1.0).

We posit that the Narrative Structures defined in Section 2 (such as The Hero's Journey) function as **Lossy Compression Algorithms**. By encoding complex social data into a pre-trained schematic (the "Monomyth" prior), the brain drastically reduces the cognitive load required to process the information. Stories are comfortable precisely because they render the chaotic world predictable, allowing the brain to minimise its energy expenditure.

When a reader recognises a story as a "Tragedy" (Freytag's Pyramid), they effectively load a specific high-level prediction model. "Good" storytelling is the optimisation of this prediction error:

1. **Boredom:** Prediction error is zero. The model is too accurate; no new information is acquired.
2. **Confusion:** Prediction error is maximal. The model fails completely; the narrative lacks coherence.
3. **Engagement:** Prediction error is minimised over time through the resolution of conflict. This represents the optimal state of Active Inference.

4.3 Narrative as Active Inference

Bouizegarene et al. [22] propose the framework of "Narrative as Active Inference." They argue that humans are agents capable of **Temporal Depth**.

A simple agent (like a thermostat) reacts only to the present state. A narrative agent (a human) projects a "Self" into a simulated past and a simulated future.

$$\text{Policy}(\pi) = \arg\min_{\pi} G(\pi) \tag{18}$$

Here, G is the expected free energy of a future path. In essence, narratives serve as a **Flight Simulator** for social existence. They allow us to test various policies (π)—such as "what are the consequences of betrayal?"—without incurring the existential risks of real-world failure.

4.4 Neural Entrainment and Coupling

The physical manifestation of this algorithmic alignment is known as **Neural Coupling**. Using fMRI analysis, Hasson et al. [23] provided empirical evidence for this phenomenon.

They demonstrated that during the narration of a compelling story, the listener's brain activity synchronises with the storyteller's brain activity. The same neural networks activate in the same sequence, with a marginal temporal delay. Crucially, this synchronisation *fails* if the narrative structure is disrupted (e.g., by randomizing the sentence order).

This provides neuroscientific support for our central thesis: Narrative Structures are not arbitrary artistic conventions. They are the standardised communication protocols—the "handshakes"—required to establish a high-bandwidth connection between two predictive brains.

5 System Architecture: The Neuro-Symbolic Blueprint

We have established that narrative structures function as optimisation algorithms (Section 2) and that archetypes function as latent centroids (Section 3). The final challenge is architectural: *How do we instantiate these theoretical models within a single computational agent?*

The current landscape of Artificial Intelligence presents a dichotomy. On one side, we have **Symbolic AI** (often termed GOFAI or "Good Old-Fashioned AI"). This approach represents knowledge using explicit logical rules and symbols (e.g., *If A then B*). It excels at long-term planning and logical consistency but lacks linguistic fluency; it cannot write a poem, but it can play chess perfectly. On the other side, we have **Neural Language Models** (such as GPT-3 [24]). These are statistical models that learn patterns from vast amounts of data. They possess high linguistic fluency but lack an underlying world

model; they can write a poem, but they often fail to maintain basic logical continuity [25].

To solve the complex problem of automated story generation, we propose a **Neuro-Symbolic Architecture**. This hybrid approach assigns specific cognitive tasks to the module best suited for them: the high-level "Plot" is handled by a symbolic planner, while the low-level "Prose" is handled by a neural probabilistic model.

5.1 The Limitations of Pure Neural Models

While Transformer-based models like GPT-3 [24] demonstrate remarkable capabilities in text generation, they fundamentally operate as probabilistic sequence predictors. Mathematically, they predict the next token (word or character) w_t based on the conditional probability distribution $P(w_t|w_{t-k}...w_{t-1})$.

Without an external control mechanism, these models suffer from two algorithmic deficits crucial to storytelling:

1. **The Horizon Problem (Wandering Plots):** Neural models are optimised to minimise entropy at the local level (the next few words). They lack a global objective function. Consequently, the narrative trajectory often drifts; the model cannot "remember" a plot goal established 500 tokens prior because its optimisation window is finite.
2. **Causal Hallucination:** The model prioritises semantic association over logical causality. For example, if the context contains the words "door" and "locked," the model might statistically predict the word "key." However, it does not check the logical state of the world to verify if the protagonist actually possesses a key. It hallucinates the solution because it fits the linguistic pattern, not the logical reality.

5.2 Architecture: The Cortex and The Broca

To mitigate these limitations, we propose a bipartite system architecture inspired by the human brain's separation of executive function (planning) and linguistic production (speech). This division of labour can be understood through the analogy of a film production:

- **The Symbolic Planner (The Director):** This module does not speak. It holds the script (The Narrative Structure), knows the plot beats, and gives instructions (e.g., "The scene must be sad" or "The hero must fail here").
- **The Neural Renderer (The Actor):** This module improvises the dialogue and action. It takes the instruction ("be sad") and translates it into a performance ("tears ran down my face").

5.2.1 The Symbolic Planner (The Cortex)

This module acts as the executive controller. Crucially, it functions as a **Structural Multiplexer**. Depending on the desired narrative genre, it loads the specific mathematical definitions established in Section 2:

- If the story is a **Tragedy**, it loads the Gaussian Envelope (Eq. 2) to monitor tension.
- If the story is a **Quest**, it loads the Gated Finite State Machine (Section 2.6) to track inventory.
- If the story is a **Thriller**, it loads the A* Pacing Constraints (Section 2.7).

While this prototype demonstrates the architecture using three distinct paradigms, the multiplexer is designed to be modular; new narrative structures can be integrated simply by adding a new case handler and its corresponding cost function.

The Planner utilises these structures to generate **Constraints**, not text. It maintains the global state S_{world} and ensures logical consistency. The explicit state of the world is stored as a set of logical predicates:

$$S_{\text{world}} = \{\text{HeroAt}(Cave), \text{Has}(Sword), \text{Enemy}(Dragon)\} \quad (19)$$

5.2.2 The Neural Renderer (The Broca)

This module is a pre-trained Transformer (e.g., GPT-3). Its role is to translate the abstract constraints provided by the Planner into natural language. This is where the **Archetypal Centroids** from Section 3 are applied. When generating dialogue for a specific character, the system imposes the **Vector Quantisation** constraint (Eq. 16). It injects the centroid vector μ_{shadow} or μ_{hero} into the context, mathematically biasing the probability distribution. This ensures that the "Actor" stays in character, penalising any generation that drifts too far from the archetypal neighbourhood.

5.3 The Generative Algorithm: Dynamic Constraint Switching

The core innovation of this proposal is the control loop detailed in Algorithm 8. Unlike standard LLM generation which relies solely on "Next Token Prediction" (an open-loop process), our architecture implements a **Closed-Loop Control System**.

This loop is the computational implementation of the **Active Inference** framework discussed in Section 4. In biological terms, the brain continuously predicts sensory input and updates its model based on prediction error. We replicate this cognitive process through three distinct phases:

Phase 1: Top-Down Prediction (The Director). The Symbolic Planner looks at the current timestamp t and the selected narrative structure (e.g., Freytag's Pyramid). It calculates the **Expected Narrative State** (S_{target}).

- *Example:* "At $t = 0.8$, the structure dictates that Tension must be High (0.9) and the Hero must be in a state of 'Defeat'."

This expectation acts as a **Prior Belief**. It is fed into the neural network not just as a text prompt, but as a set of Logit Biases and constraint parameters.

Phase 2: Bottom-Up Generation (The Actor). The Neural Renderer samples a candidate sequence x based on the prompt. This process is stochastic; the "Actor" improvises a scene. This generated text represents the **Sensory Data**—the actual realisation of the story in the environment.

Phase 3: Surprisal Minimisation (The Critic). The system compares the Top-Down Prediction (S_{target}) with the Bottom-Up Sensory Data (x) to calculate the **Free Energy** (or Prediction Error).

$$\mathcal{F} = \mathcal{L}_{\text{structure}}(S_{\text{target}}, x) + \mathcal{L}_{\text{logic}}(S_{\text{world}}, x) \tag{20}$$

This error metric asks two questions:

1. *Structural Fit:* Did the Actor play the scene with the correct emotion? (Calculated via Sentiment Analysis).
2. *Logical Fit:* Did the Actor contradict established facts? (Calculated via NLI checking).

If the Error $\mathcal{F}$ exceeds a tolerance threshold ϵ, the system triggers a **Re-weighting Event**. It identifies the tokens that contributed most to the error (e.g., words associated with happiness in a sad scene) and applies a negative penalty to their logits. The model then regenerates the sequence. This cycle repeats until the "Surprisal" is minimised, mathematically guaranteeing that the generated prose aligns with the structural intent.

Algorithm 8: Neuro-Symbolic Constraint Loop

Input: GenreType T, Archetypes $\mathcal{A}$, StartNode S_0
Output: Coherent Narrative History H

$State \leftarrow S_0$;
$History \leftarrow$ "Once upon a time...";
while $State \neq EndNode$ **do**
  // 1. Director Phase: Select Active Math Model
  $Constraint \leftarrow \emptyset$;
  switch *GenreType* **do**
    case *Tragedy* **do**
      $TargetTension \leftarrow$ GaussianEq(Time) // Eq. 1
      $Constraint \leftarrow TargetTension$;
    end
    case *Quest* **do**
      $Reqs \leftarrow$ GetPrerequisites($State$) // Sec 2.6
      $Constraint \leftarrow Reqs$;
    end
    case *Thriller* **do**
      $Pace \leftarrow$ AStarHeuristic($State$) // Sec 2.7
      $Constraint \leftarrow Pace$;
    end
  end
  // 2. Actor Phase: Neural Sampling
  $Prompt \leftarrow$ Format($History, Constraint, \mathcal{A}$);
  $Candidate \leftarrow$ GPT3.Sample($Prompt$);
  // 3. Verification Phase (Free Energy Minimisation)
  $Error \leftarrow$ CalculateLoss($Candidate, Constraint$);
  if $Error < \epsilon$ ***and*** CheckLogic($Candidate$) **then**
    $History \leftarrow History + Candidate$;
    $State \leftarrow$ UpdateWorldModel($State, Candidate$);
  end
  else
    // High Surprisal: Reject & Penalise
    ReweightLogits(FailureTokens, Penalty = $Error$);
    continue // Regenerate with new weights
  end
end

By constraining the immense, chaotic search space of a Neural Language Model with the rigorous graph topology of Computational Narratology, we can achieve systems that display both the creative fluency of a human writer and the structural coherence of a master storyteller.

6 Discussion and Future Work

This research has undertaken a formal deconstruction of storytelling, reframing it not as an artistic abstraction but as an algorithmic necessity for cognitive processing. Through the course of this paper, we have mapped qualitative literary structures to rigorous computer science paradigms:

- **Signal Processing:** Modelling Freytag's Pyramid as a tension envelope.
- **Graph Theory:** Modelling the Hero's Journey as a directed graph traversal.
- **Search Heuristics:** Modelling pacing (Save the Cat) as an A* optimisation problem.

By viewing these structures through the lens of the **Active Inference** framework, we argue that narratives may function as optimisation heuristics designed to minimise entropy in the human brain. In this view, a story serves as a tool allowing a predictive mind to compress complex causal information into a manageable format. The **Director-Actor** architecture proposed in Section 5 attempts to operationalise this theory, exploring the shift of story generation from an "Open-Loop" process (unsupervised probability) to a "Closed-Loop" process (supervised intent).

This perspective offers noteworthy implications for Artificial Intelligence. It suggests that simply increasing the parameter count of Neural Networks (scaling up models like GPT-3) may not spontaneously result in "true" narrative understanding. As our analysis posits, narrative coherence appears to be a **Structural Property**, not merely a statistical one. Consequently, models that predict words based on probability alone may continue to struggle with long-term causality without the guidance of an external Symbolic Planner.

6.1 Future Research Directions

To move closer to an AI that can truly co-create with humans, future work must focus on the practical implementation of the Neuro-Symbolic architecture proposed in Section 5. We identify multiple critical areas for development:

1. **Prototype Code Implementation:** The immediate next step is to translate the theoretical algorithms proposed in this paper into a functional software library. We propose a Python-based implementation utilising **PyTorch** [26] for the tensor operations (Archetypal Vector Quantisation) and **NetworkX** [27] for the topological modelling (Narrative Graphs). Creating a "playground" environment where users can toggle different narrative structures (e.g., swapping a Tragedy module for a Hero's Journey module) would provide essential empirical data on the efficacy of these constraints.

2. **Optimisation of the Control Loop (Latency):** The "Generate-and-Evaluate" cycle proposed in Algorithm 8 introduces significant computational latency. Because the "Actor" may be forced to regenerate scenes multiple times to satisfy the "Director's" constraints, the system is slower than a standard LLM. Future research should investigate **Speculative Decoding** or **Guidance Distillation** techniques to minimise the number of rejection steps required, effectively teaching the neural network to predict the constraint violations before they occur.
3. **Dynamic Graph Generation:** Currently, our proposed Symbolic Planner relies on static, pre-defined templates (e.g., forcing every story to follow the Hero's Journey). This limits creativity. A more advanced system should utilise **Crowdsourced Plot Graphs**, as proposed by Li et al. [28]. By mining data from thousands of stories, the AI could learn to construct *new* graph topologies dynamically, effectively inventing new narrative structures rather than simply filling in the blanks of old ones.
4. **Commonsense World Modelling:** The "Symbolic Planner" requires a robust understanding of cause and effect to maintain the state S_{world}. To achieve this, we propose integrating **Knowledge Graphs**, specifically **COMET** (Commonsense Transformers) [29]. COMET is a system designed to understand implied causality (e.g., knowing that "If X buys a sword, X loses money" and "X now has a weapon"). Integrating this would allow the Director to infer unstated preconditions without requiring manual hard-coding.
5. **Refining the Error Function ($\mathscr{F}$):** The success of our architecture depends entirely on the accuracy of the Surprisal metric ($\mathscr{F}$). If the system cannot accurately detect that a scene is "happy" when it should be "sad," the feedback loop fails. Future work must develop more nuanced **Style Classifiers** and **Logic Checkers** that can handle the subtlety of literary subtext, rather than relying on crude sentiment analysis.

6.2 Conclusion

In conclusion, this research suggests that the path to computational creativity may not rely solely on the brute force of modern supercomputers, nor on the rigid rules of the past, but rather on their synthesis. We propose that a promising direction lies in the marriage of the old and the new: the ancient, structured wisdom of the storyteller (Symbolism) and the modern, statistical power of the neural network (Connectionism). By striving to build machines that understand the *shape* of a story—machines designed to act as both Director and Actor—we aim to explore how AI might eventually better approximate the human capacity for narrative.

References

[1] K. Friston, "The free-energy principle: A unified brain theory?" *Nature Reviews Neuroscience*, vol. 11, no. 2, pp. 127–138, 2010.

[2] T. Todorov, *Grammaire du Décaméron*. Mouton, 1969.

[3] I. Mani, *Computational Narratology*. Walter de Gruyter, 2012.

[4] G. Freytag, *Die Technik des Dramas*. Hirzel, 1863.

[5] M. O. Riedl and R. M. Young, "Narrative planning: Balancing plot and character," *Journal of Artificial Intelligence Research*, vol. 39, pp. 217–268, 2010.

[6] J. Gardner, *The Art of Fiction: Notes on Craft for Young Writers*. Vintage Books, 1983.

[7] Aristotle, *Poetics*. Penguin Classics, 1996, Original work ca. 335 BCE.

[8] S. Field, *Screenplay: The Foundations of Screenwriting*. Delta, 1979.

[9] M.-L. Ryan, *Possible Worlds, Artificial Intelligence, and Narrative Theory*. Indiana University Press, 1991.

[10] D. Wells, *7 point story structure*, Lecture Series at LTUE Symposium, Available via YouTube/Writing Excuses, 2013.

[11] J. Campbell, *The Hero with a Thousand Faces*. Pantheon Books, 1949.

[12] C. Vogler, *The Writer's Journey: Mythic Structure for Writers*. Michael Wiese Productions, 2007.

[13] M. Mateas and A. Stern, "Façade: An experiment in building a fully-realized interactive drama," in *Game Developers Conference*, vol. 2, 2003.

[14] B. Snyder, *Save the Cat! The Last Book on Screenwriting You'll Ever Need*. Michael Wiese Productions, 2005.

[15] C. G. Jung, *The Structure and Dynamics of the Psyche* (Collected Works of C.G. Jung, Vol. 8). Pantheon, 1960.

[16] C. G. Jung, *The Archetypes and the Collective Unconscious* (Collected Works of C.G. Jung, Vol. 9 Part 1). Pantheon, 1959.

[17] R. A. Jones and L. Gardner, *Narratives of Individuation*. Routledge, 2019, ISBN: 978-0-429-51470-8.

[18] C. S. Pearson, *Awakening the Heroes Within: Twelve Archetypes to Help Us Find Ourselves and Transform Our World*. HarperSanFrancisco, 1991.

[19] D. Bamman, B. O'Connor, and N. A. Smith, "Learning latent personas of film characters," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013, pp. 352–361.

[20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[21] A. Clark, "Whatever next? predictive brains, situated agents, and the future of cognitive science," *Behavioral and Brain Sciences*, vol. 36, no. 3, pp. 181–204, 2013.

[22] N. Bouizegarene, M. J. Ramstead, A. Constant, K. J. Friston, and L. J. Kirmayer, "Narrative as active inference: An integrative account of the functions of narratives," *PsyArXiv Preprints*, 2020.

[23] U. Hasson et al., "Neural coupling and storytelling," *Proceedings of the National Academy of Sciences*, vol. 107, no. 32, 2010.

[24] T. B. Brown et al., "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[25] G. Marcus, "The next decade in ai: Four steps towards robust artificial intelligence," *arXiv preprint arXiv:2002.06177*, 2020.

[26] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[27] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, 2008, pp. 11–15.

[28] B. Li et al., "Story generation with crowdsourced plot graphs," in *AAAI Conference on Artificial Intelligence*, 2013.

[29] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi, "Comet: Commonsense transformers for automatic knowledge graph construction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4762–4779.