Epic-Level Text Generation with LLM through Auto-prompted Reinforcement Learning

1st Qianqian Qi
The University of Auckland
qqi518@aucklanduni.ac.nz

2nd Lin Ni
The University of Auckland
l.ni@auckland.ac.nz

3rd Zhongsheng Wang
The University of Auckland
zwan516@aucklanduni.ac.nz

4th Libo Zhang
The University of Auckland
lzha797@aucklanduni.ac.nz

5th Jiamou Liu
The University of Auckland
jiamou.liu@aucklanduni.ac.nz

6th Michael Witbrock ⋆
The University of Auckland
m.witbrock@aucklanduni.ac.nz

2024 International Joint Conference on Neural Networks (IJCNN) | 979-8-3503-5931-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/IJCNN60899.2024.10650358

***Abstract*—In an era where the capabilities of large language models (LLM) like ChatGPT are transforming digital communication, the challenge of directing these tools to create extensive, coherent narratives on an epic-scale has emerged as a critical frontier. This study introduces a novel methodology that fuses the spontaneous story generation of LLMs with the precision of auto-prompted reinforcement learning for crafting epic-scale, coherent narratives. Our approach starts with generating a skeletal outline, followed by iterative expansion, and blending operations for maintaining structural coherence in long-form content. To train the reinforcement learning model efficiently, we introduce an environment simulator that leverages a database of historical LLM interactions, circumventing the limitations of direct LLM interactions. This method enhances the decision-making process of the RL agent, enabling more effective prompt selection and narrative flow in extended texts. We validate its effectiveness through experiments, demonstrating the model's ability to generate structured, narrative-driven text, thereby setting a new pathway towards AI-driven, large-scale storytelling.**

***Index Terms*—GPT-3.5, story generation, reinforcement learning, auto-prompt**

I. Introduction

The realm of text generation has witnessed significant advancements, enabling Large Language Models (LLMs) like ChatGPT to generate extensive pieces of content spanning a few thousand words. However, the pursuit of achieving epic-level text generation with arbitrary long text remains a substantial challenge. The first challenge is *managing the story's narrative structure*: While an LLM is capable of generating vast amounts of text, guiding the model to create text that adheres to a predetermined plot or leads to a logical conclusion presents challenges [1]. User prompts can guide the model, but they have limited power, and the model may deviate from the desired narrative when the text becomes longer. The second challenge is *maintaining consistency*: Long text often requires maintaining a consistent narrative over thousands of words, which can be challenging for an LLM [2]. While a human writer naturally keeps these elements in mind while writing, an LLM has no inherent understanding of the underlying plot or characters, making it difficult to keep a long narrative consistently in line with the text's established elements [3]. The third challenge is *evaluation*: it's difficult to measure how well an LLM performs in long text generation. Traditional automatic evaluation metrics like BLEU [4], ROUGE [5], or even more recent ones like BERTScore [6], may not capture all the differences of a well-written long text [7]. Human evaluation is more reliable but costly and time-consuming.

The main motivation of this paper is to address the challenges above by developing a novel framework that enhances the capability of LLMs to generate long texts with consistent narratives and structured storylines.

1) Develop a strategy for generating and refining storylines: the first issue is to generate the storyline from the provided title and subsequently refine it as the story generation progresses. assessing plot conflicts and iteratively refining the storyline to enhance consistency.
2) Optimizing LLMs in text Generation for story plots: The second issue lies in developing algorithms to guide the LLM effectively, ensuring the coherent and purposeful creation of narrative content in alignment with a predefined storyline and preceding plot.
3) Evaluating Long Text Requirements: The third aspect is evaluating whether the generated text fulfills the requirements of a long text. This involves assessing factors such as narrative structure, plot coherence, and overall text generation effectiveness. Evaluating the text against these criteria helps determine the success of the long text generation process.

Navigating the complexities of text generation demands a strategic approach that ensures complete storyline and consistent story. Reinforcement learning (RL) provides a potent and promising approach to guide a LLM to maintain consistency [8]. In RL, an agent learns from the reward feedback based on its actions. This feedback loop can help guide a LLM's text generation according to requirements such as maintaining high consistency level. Moreover, RL can balance exploration (trying out new ways of text generation) and exploitation (using known successful methods) [9]. This can be crucial in striking a balance between coherence and diversity in

⋆ Corresponding Author.

the generated text while maintaining consistency. Thirdly, another key benefit of RL is its ability to handle delayed rewards, which is crucial in long text generation. In a long text generation, the impact of a decision may not be seen immediately but could affect the quality of the text much later. RL is well-suited to optimize for such scenarios.

In this paper, we propose an auto-prompted RL approach for epic-level text generation using GPT-3.5 [1]. In a nutshell, our approach generates a skeleton outline first, and then iteratively expands each chapter using RL agent to select suitable prompts for GPT-3.5. More specifically, our contributions are three-fold: **(1)** We design an auto prompt strategy to generate text that aligns with both the established storyline and the predefined plot. This ensures textual continuity by guiding the LLM towards predefined storyline, preventing the repetition of plot generation and ensuring consistency in role events. **(2)** We model the epic-level text generation process as a Markov Decision Process (MDP), providing a formal framework for solving the problem with RL. We put forward an automated process for addressing challenges in epic-level text generation by employing RL for story plot progress controller. **(3)** Recognizing the time and cost challenges associated with RL training in a real interactive environment, we introduce an Environment Simulator. Leveraging historical interactions, this simulator generates simulated responses based on the current state and action, enabling efficient expression of simulated action replies. This innovation mitigates the lengthy training duration associated with traditional RL methods.

In our experiments, we used the Proximal Policy Optimization (PPO) algorithm for reinforcement learning, creating a comprehensive environment with states, actions, and rewards as detailed in Chapter IV-D. We employed GPT-3.5, fine-tuned for versatility across text genres. The RL policy, initiated with a prompt template, updated parameters based on chapter-wise rewards. Evaluation included assessing average rewards for extended story generation. We compared our approach with RecurrentGPT, a model leveraging interactive storytelling and emulating long-short-term memory architectures, serving as a benchmark for our work. Our experiment results show that the proposed approach is effective in controlling LLM in generating long and consistent text with a strong sense of narrative structure. Furthermore, we define the evaluation metrics to guide the text generation with the constraint of a predefined structure and guaranteeing a satisfying conclusion.

II. Related Work

Many existing work has been conducted on employing AI in the domain of text generation. In the project TALE-SPIN, Meehan [10] was the first to utilize AI for the automatic generation of stories. Unlike story grammars, TALE-SPIN focused on characters' desires and intentions, employing AI problem-solving techniques to fulfill these objectives. AI planning in story generation involves providing an initial state and a goal, with a reasoner inferring actions to lead the initial state to the story goal, and a directing process may be employed to improve the quality of the generated story. In [11], they develop Plotter, an introduced computational tool, is capable of generating story plots using the plot fragments from Plotto along with their corresponding instructions. The success of Seq2Seq models in various natural language processing (NLP) tasks motivated researchers to explore their application in generating complete stories [12]. The work in [13] devised a story generator by combining two readily available systems. This novel approach allows the generator to produce stories when provided with a sequence of separate, unrelated short descriptions. Text samples generated by the GPT-2 [14] demonstrate that these PLMs can generate text that rivals human writing. In [15], they study on write intermediate plot structure that connects the given prompt with the final generation of a story. They fine-tune BART [16] using two sets of training data. The first set consists of prompts and extracted plot structures that serve as a standard reference. The second set includes prompts combined with plots and their corresponding stories.

Reinforcement learning has been explored as another approach in the domain of control language models. The work in [17] extended the use of reinforcement learning to controllable story generation. They introduced a reward-shaping technique that generates intermediate rewards at each timestep, and these rewards are subsequently incorporated into a language model through back-propagation. This approach effectively directs the generation of plot points toward a specified objective or goal. However, when it comes to long text generation, it operates in an open domain where we don't specify a particular object or direction for the text. GPT-based agents like AutoGPT [2] are significant in the realm of computer-assisted writing systems [18]. The current leading systems in this field, as pointed out in [19], primarily concentrate on offering localized editing suggestions and generally regard Large Language Models (LLMs) as "black boxes". This means these systems use the outputs of the models without fully understanding their internal workings or decision-making processes. RecurrentGPT [20] operates by receiving and updating various inputs such as previous content, short-term and long-term memory of the content, and content outlines at each time step. It initially generates the first paragraph and suggests potential continuation plans, but relies on the writer to choose, modify, or create new plans. This method is not fully autonomous and lacks clear termination conditions, leading to uncertainty about when the text should conclude.

III. Problem Definition

In this project, epic-level text generation pertains to producing text based on a given title I via a automated text generation system denoted as $Generator$.

$$\begin{aligned} E &= Generator(I) \\ Generator(I) &= Controller(LLM(I, Pro_i)) \end{aligned} \tag{1}$$

[1]https://platform.openai.com/docs/model-index-for-researchers

[2]https://github.com/Significant-Gravitas/Auto-GPT

TABLE I
QUANTITATIVE AND QUALITATIVE EVALUATION METRICS FOR EPIC-LEVEL TEXT GENERATION

Metrics	Description
number of paragraphs	Total number of paragraphs in the currently generated text.
length of text	Total number of words in the currently generated text.
number of simple sentences	Total number of sentences with fewer than 15 words.
number of low-frequency words	Number of occurrences of low-frequency words (as counted using the wordfreq library).
coherence score	Evaluation of results conducted by ChatGPT (on a scale of 0-5).
consistency score	Evaluation of results conducted by ChatGPT (on a scale of 0-5).
number of recall nodes	Number of significant occurrences that convey the idea of specific points.

Here, E represents the resulting story. The aim of this study is to design $Generator$, which takes the form of an LLM controller. The controller utilize the LLM in conjunction with the generated prompt Pro_i to produce content accordingly. Firstly, we generate a storyline with I. To enhance specificity, we employ a top-level outline $s_0 = Outliner(I)$ as the storyline. As the storyline progresses, a detailed sub-level outline s_i is developed, and content is generated in alignment with the high-level outline.

We illustrate the example using the provided title "Snow White", as depicted in Figure 3. The top-level outline is generated as follows: "• **Chapter 1 The Evil Queen:** The story starts with the introduction of the villain, the Evil Queen. She is obsessed with her own beauty and jealous of anyone who is more beautiful than her. She learns about Snow White who is the fairest of them all and decides to get rid of her. • **Chapter 2 The Escape:** Snow White is aware of the evil queen's wicked plan and escapes into the forest. She eventually meets the seven dwarves who offer her shelter and protection. They become friends and Snow White helps them in return. • **Chapter 3 The Huntsman's Failed Mission:** ...".

Concurrently, we iteratively refine the outline as the storyline progresses. For instance, once Chapter 1.1 is generated, the system may adjust subsequent outlines, such as Chapter 1.2, Chapter 2, and so forth, in response to the evolving content of the story. We employ several features strategically incorporated to indicate the progression of the ongoing storyline, as described below. Each chosen feature contributes a unique dimension to the narrative, serving as a dynamic indicator of the story's unfolding complexity.

$$\begin{aligned} Pro_i &= \text{PromptGenerator}(l_{i-1}, p_{i-1}, n_{i-1}, s_{i-1}) \\ G_i &= \text{LLM}(Pro_i) \\ l_i &\mathrel{+}= Len(G_i) \\ n_i &= \{n_0, \ldots, n_{i-1}, \text{LLM}(G_i)\} \\ p_i &\mathrel{+}= NumLowFreq(G_i) \\ s_i &= \text{Outliner}(l_i, p_i, n_i, s_{i-1}) \\ E &= \{G_0, G_1, \ldots, G_i\} \end{aligned} \tag{2}$$

An *epic-level* story is a long and expansive narrative that captivates readers with its length (l), intricate word-building (p), plot (n) and story outline (s). The length l represents the word size of the narrative. Epic-level texts typically unfold over a considerable amount of words. p utilize the identification of low-frequency words as a discriminative factor for determining the writing level [21]. The plot n refers to the sequence of events that make up the story. In an epic-level text, the plot is typically complex, involving multiple intertwined storylines, a large number of significant characters, and various major and minor conflicts. The narrative structure s refers to the outline that shapes the storyline of the text. Each of these features contributes to the epic scale, complexity, and richness of the narrative, helping to captivate and engage readers.

In the LLM controller, a control mechanism is required to overcome a primary challenge with current LLMs: their inability to process long text simultaneously. For a more cohesive comprehension of the text, it's necessary to plan structure and arrange the generated text, which allows for a thorough understanding of the current state of the writing. Hence, we require a collection of pre-defined prompt templates $Pro = \rho_1, \rho_2, ...$, which come with slots for integrating context information $\{l, p, n, s\}$, obtained from the existing narrative. So, the approach involves choosing the suitable prompt for LLM based on the context information and generating text in an iterative manner. Pro_i is the selected prompt in timestep i, and G_i is the corresponding response from LLM. E is consist of G_i. s_0 is created based on the provided title I, and l_0, p_0 are initialized to 0. Initially, n_0 is set to be empty. The variable i denotes the step of the $Controller$, starting from 1. The above procedure is formulated as Equation 2.

IV. METHOD

A. Paradigm

Following standard terminologies in RL, an MDP is represented as a tuple (S, A, r, γ, P), where S denotes the set of states, A represents the set of actions, $r : S \times A \to \mathbb{R}$ defines the reward function, γ is the discount factor, and P refers to the transition function.

In our specific task of generating epic-level text, we define the state space as $S = \mathbb{R}^k$, which corresponds to a k-dimensional vector space used for representing the snapshot of the narrative context $\{l, p, n, s\}$. The action space A consists of interactive prompts that are employed iteratively to control LLM. The reward function $r = func(E)$ is determined by applying conventional natural language evaluation metrics to assess the quality of the generated text. Thus, our objectives comprise two essential components: identifying optimal prompts for LLM to generate the storyline's plot content and ensuring the coherence of the narrative via storyline. We aim to

produce epic-level text that maximizes the cumulative reward, $\sum_{t=0}^{T} \gamma^t r(s^t, a^t)$, where $T \in \mathbb{R} \bigcup \infty$ represents the maximum number of steps, based on our specified metrics.

After having the MDP formulation, we need to then develop a method to achieve our goal, by finding the optimal policy to the MDP $\pi : S \rightarrow \triangle(A)$, a mapping from the states to the distribution over actions. We apply the PPO algorithm [22], a deep RL algorithm that parameterizes the policy π with a parameter θ. The optimization objective of our problem becomes: $\max_\theta \mathbb{E}_{\tau \sim \pi_\theta, \eta}[r(\tau)]$ where η is the initial state distribution, π_θ is the parameterized policy, τ is the sequence of state-action pairs generated by π_θ under the initial state distribution, and we define the $r(\tau)$ as the cumulative reward given by trajectory τ.

Storyline Refinement: Upon completing a chapter, we initiate a comprehensive review of the current outline, deciding whether to refine or rewrite subsequent chapters. This refinement process involves evaluating the coherence and thematic consistency of the narrative. The function $Outliner$ involves adjustments and improvements to the current outline, ensuring a more polished and coherent storyline for the next chapter. This dynamic process contributes to the evolution and enhancement of the overall narrative.

Consistent Story plot content generation: Each chapter commences with content generation, and based on the current content and previous plot, the RL method selects actions, such as finishing the story, writing the next chapter, rewriting current content, or extending current content. These actions are contingent upon the current text features l, p, n, s. In this process, $PromptGenerator$ is used to choose optimal prompt while controlling LLM.

In conclusion, our proposed paradigm employs RL to iteratively refine the storyline and control the LLM for stoty plot text generation. The writing state is represented by feature values from structured text, providing a snapshot of the narrative context. The RL agent selects an optimised prompt based on this state, populates it with context, and feeds it to the LLM. RL evaluates the generated text, refining future selections. This approach optimizes long story generation, aligning task goals with RL rewards and feature-based state representation.

B. Long Texts Evaluation

We developed specific metrics for evaluating long text generation success, including narrative structure, plot coherence, and text generation effectiveness. Key metrics like length, consistency, and coherence help gauge the quality of long texts. A 'draft' strategy optimizes text construction, which undergoes automatic evaluation involving statistics and scoring. Certain factors, such as overly simple sentences, can negatively impact the score, while others may increase it.

The table's top four metrics can also be determined automatically. However, we want to use the text assessment feature on ChatGPT to assess the final two criteria (coherence and consistency). We send the text to ChatGPT, which response with an evaluation score (from 0 to 5) and an explanation of it. ChatGPT has a finite number of Tokens per API call, which is insufficient to support evaluating all currently generated texts at once. Consequently, we developed two lists for directly assessing the coherence and consistency of text parts, storing interim results for later computations. This contrasts with automatic metrics, which necessitate aggregating all long-form generated texts.

C. Environment Simulator

Using a real interactive environment to train reinforcement learning models can be both costly and time-intensive. As an example, making an API request to GPT-3.5 for a response typically takes approximately 20 seconds. In our text generation process, one episode may require over 100 API requests. Considering that traditional RL training often involves millions of timesteps to achieve a satisfactory policy, the training duration can extend beyond several months. As a result, we adopt the concept presented in the Environment simulator mentioned in [23]. The simulator is developed based on historical interactions as database. By taking the current state s^t and action a^t as inputs, the simulator generates a simulated response. This method enables us to express the simulated action reply as $Sim(s^t, a^t)$, where the precise definitions of $Sim(\cdot)$ may differ depending on the specific reinforcement learning (RL) models used.

D. Epic-level Story Generation Prototype

Our research resulted in a prototype for long story generation, using the mentioned methodologies to create contextually consistent narratives. The prototype's performance, evaluated with our new metrics, showcases its ability to generate appropriate long texts. The design and performance specifics of this prototype are detailed in this final section.

a) Framework: The framework, illustrated in Figure 1, employs PPO as the chosen reinforcement learning algorithm for policy parameter updates which is widely used in Natural Language Processing (NLP) [24].

b) Action Definition: To ensure consistency, coherence, detail, length, minimal repetition, and reader engagement in story generation, we have defined the following actions: ***plan, draft, revise, augment, expand, resolve, and terminate***. The ***plan*** action creates outlines for chapters and sub-chapters, ensuring book consistency [25]. The ***draft*** action generates controlled content for ongoing chapters [26]. ***Revise*** restructures outlines or improves existing drafts for text quality [27]. ***Augment*** incorporates dialogue and background descriptions for increased engagement. ***Expand*** extends drafts while maintaining coherence [28]. ***Resolve*** integrates the current draft into the main body, and ***terminate*** concludes the training process.

c) State Definition: To support the desired functionalities of the actions, our state includes: ***Main Novel*** (E), ***Novel Outline*** ($outline$), ***Current Plan*** ($plan_{current}$), ***Current Draft*** ($draft_{current}$), ***Plot Events*** ($events$). The $outline$ is structured as a dynamic spanning tree, with nodes representing pairs of chapter titles and brief summaries. By utilizing a depth-first search (DFS) on this outline tree, we generate paragraphs as leaf nodes for each chapter. Through dynamic programming,

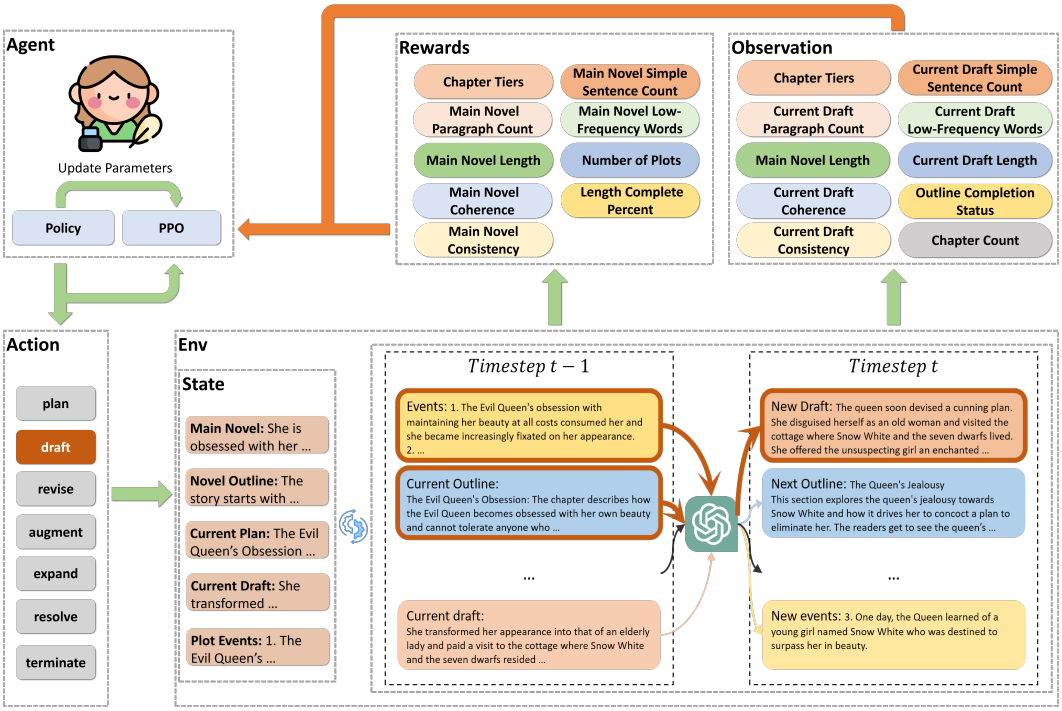

Fig. 1. The framework consists of three key components: the agent, with policy and Proximal Policy Optimization (PPO) sections, and the environment (Env). The agent makes decisions and chooses actions based on conditions. The PPO guides these actions using past learnings. Env interacts with GPT-3.5 and stores state information. The agent bases its next action on rewards and observations like paragraph count and text length.

we can expand the tree by recursively adding new nodes to the current chapter node. This approach maintains the outline as the novel's backbone, providing both organization and flexibility for incorporating details and exploring deeper content. The $plan_{current}$ represents the currently traversed node during the depth-first search. It contains a title and a summary of a chapter that have yet to be resolved, serving as a plan for generating several paragraphs as the leaf of the node. The $draft_{current}$ consists of generated paragraphs for $plan_{current}$, temporarily cached for potential quality improvements through actions such as revision and expansion, as defined in Action Definition. Finally, the $events$ capture a sequence of significant events that have occurred in E. This serves as a brief memory of previous chapters, ensuring consistency and facilitating the smooth development of new plots.

d) Capture Observation: To capture the features of the writing state and represent a snapshot of the narrative context, we have designed a 10-dimensional vector as observations. These manually created features include the following components:

- **Chapter Count**: Measures the number of nodes in the $outline$, representing the count of chapters at all levels.
- **Chapter Tiers**: Indicates the depth of the $outline$, representing the hierarchical structure of chapters.
- **Outline Status**: An indicator of whether $plan_{current}$ is empty, which implies that either there is no $outline$ or all chapters in $outline$ have been resolved.
- **Current Draft Paragraph Count**: Measures the number of paragraphs in $draft_{current}$.
- **Current Draft Length**: Considers the word count of $draft_{current}$.
- **Current Draft Coherence**: Assesses the logical and consistent flow of ideas in $draft_{current}$.
- **Current Draft Consistency**: Measures the uniformity and adherence to established rules within $draft_{current}$.
- **Current Draft Simple Sentence Count**: Considers the number of simple sentences in $draft_{current}$.
- **Current Draft Low-Frequency Words**: Reflects the usage of unique vocabulary in $draft_{current}$.
- **Main Novel Length**: Considers the word count of E.

$$
\begin{aligned}
r &= r_1 + r_2 + r_3 \\
r_1 &= 0.1 \times tier_{count} + 0.1 \times para_{count} \\
&+ 0.01 \times length - 0.1 \times simp\text{-}sent_{count} \\
&+ 0.2 \times (coh + cons) + 0.1 \times plot_{num} \\
&+ 0.1 \times low\text{-}freq\text{-}word_{count} \\
&\textit{when one top level chapter is completed:} \\
r_2 &= 200 \times (length_{percent} - 0.5) \\
&\textit{when all chapters are completed:} \\
r_3 &= (length - 10000)/100
\end{aligned} \tag{3}
$$

e) Rewards Calculation: Rewards are calculated based on the state. Similar to the observations, we have designed manually created factors, but based on the main novel E instead of $draft_{current}$. The reward function is designed as Equation 3 Here, the reward is composed of the following components: ***Chapter Tiers*** ($tier_{count}$), ***Main Novel Paragraph Count*** ($para_{count}$), ***Main Novel Length*** ($length$), ***Main Novel Coherence*** (coh), ***Main Novel Consistency*** ($cons$), ***Main Novel Simple Sentence Count*** ($simp\text{-}sent_{count}$), ***Main Novel Low-Frequency Words*** ($low\text{-}freq\text{-}word_{count}$), ***Number of***

Plots ($plot_{num}$), ***Length Complete Percent*** ($length_{percent}$). The "Number of Plots" is the count of *events*, while the "Length Complete Percent" represents the progress made in completing the overall length of the novel as a percentage.

We evaluate the novel's quality by considering various weighted factors, resulting in an overall score. The factor weights in the reward calculation are subjective and adjustable based on specific needs or preferences.

f) Training Process: As describes in algorithm 1, the initiation of the agent's training process involves the formulation of a top-level outline and the initial action plan. Following the selection of an action, a prompt is chosen and combined with state components. GPT-3.5 serves as a proxy for these actions, as illustrated in Figure 1. Subsequently, the newly constructed prompt, filled with relevant state information from Timestep $t-1$, is transmitted to the Language Model Proxy GPT-3.5, which generates an adaptive response. The environment then updates its state based on the action and text, preparing for future utilization at Timestep t. These new observations and rewards guide the prediction of future actions using the PPO algorithm to update policy parameters.

Algorithm 1 Training Process

Initialize:
Construct the RL policy $\pi : S \to \triangle(A)$ parameterized by θ using PPO. Initialize environment with predefined state, action space, and reward function.
Set GPT-3.5 as LLM proxy, story title I as initial state s^0 and timestep $t = 1$.
Define prompt templates $Pro = \rho_1, \rho_2, ...$ corresponding to A.
repeat
    Step 1: Action Selection Policy π selects action a^t (a^1="plan").
    Step 2: Prompt Construction Construct a new prompt ρ'_t with components in s^{t-1}, from the prompt template ρ_t corresponding to action a^t.
    Step 3: Perform Action Send ρ'_t to LLM, receive response G_t, and update state s^t based on a^t and G_t.
    Step 4: Update Policy Calculate rewards $r^t(s^t, a^t)$, update policy parameters θ using PPO to $\max_\theta \mathbb{E}_{\tau\sim\pi_\theta,\eta}[r(\tau)]$.
    $t = t + 1$.
until termination condition

For instance, when generating a new draft (highlighted in rust color in Figure 1), we incorporate the $plan_{current}$ and *events* to form the prompt sent to GPT-3.5: "Your task is to develop specific and detailed content based on the given outline and event information list, with the requirements: ...".

We use the GPT-3.5 API for environment simulation, generating a long story titled "Snow White" with 10,000 calls. These responses are saved and used for the remaining 990,000 calls. During training, we search for similar actions in the database before using the API. If found, we select one of the top three corresponding responses randomly, otherwise, we call the API.

V. Experiments

To assess the efficacy of our approach, we devised and executed a series of experiments. For the training phase, we opted for the story title "Snow White". During the testing phase, we produced six stories and conducted a comparative analysis with the baseline model, considering factors such as text length, structural coherence, and adherence to the story plot.

A. Experimental Setup

In our experiments, we select to use Proximal Policy Optimization (PPO), a reliable and efficient reinforcement learning algorithm. The environment we have constructed involves defining states, actions, and rewards as described in Chapter IV-D. We utilized a state-of-the-art language model called GPT-3.5. This model has undergone fine-tuning using a substantial corpus of diverse text genres to ensure its versatility and applicability across a wide range of domains. While we specifically used GPT-3.5 for our experiment, it's worth noting that other language models can also be employed once their APIs are available, allowing for further exploration and comparison in future studies.

Experiment Procedure For each chapter and sub-chapter of the text, a reward system was designed, with the RL policy updating its parameters in response to these rewards. We implemented a callback: Stop Training On No Model Improvement, which would stop the training process if there was no observable enhancement in performance. This training procedure was repeated for predefined episodes or timesteps. In order to save training time, we developed a database to store the prompts and corresponding results from the OpenAI API responses. After saving the best model, we incorporated an evaluation process. This evaluation assessed the average reward for each action in extended story generation. During the training phase, we utilized a single story title, while in the testing phase, we incorporated a collection of story titles.

Baseline Comparisons We have observed RecurrentGPT, an innovative text generation model that leverages interactive storytelling. By employing state-of-the-art language models like ChatGPT, it emulates long-short-term memory architectures, facilitating the generation of extensive textual content while maintaining contextual coherence. This aligns well with our research, and we are contemplating its utilization as a baseline for our comparative analysis.

B. Results and Discussion

As show in Table II, Auto-Prompted RL generates longer, coherent narratives without recurrent human-machine interaction or reliance on short-term memory summaries. It ensures a complete story structure with a clear ending and a consistent storyline, unlike RecurrentGPT, which depends on human-provided cues at each step. The aforementioned analysis reveals that our method yields novel excerpts with a much greater volume of text. This achievement represents a substantial breakthrough in overcoming the token constraints imposed by ChatGPT's API, thereby facilitating the generation of extensive textual content. Additionally, as elucidated earlier, our generated novels possess purposeful conclusions, thereby enhancing the overall coherence of the narrative, as opposed to relying on an infinite and meandering generation of stories based on the summarization of long-short term memory.

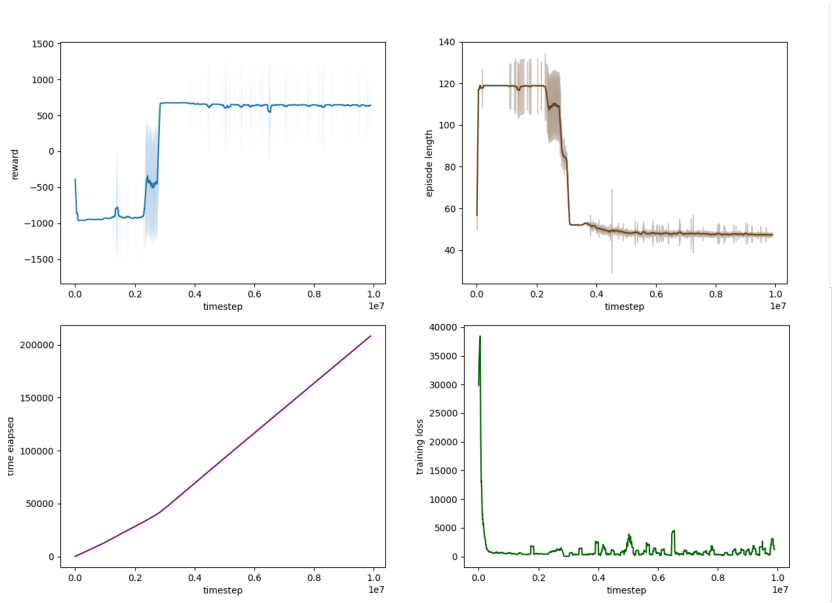

Fig. 2. In the top left graph, the reward initially starts around -400 and reaches 700 within 10 million timesteps. It stabilizes after 3 million steps at a higher value, indicating successful training. The top right graph shows a decrease in the number of steps required for completing one episode, from around 120 to approximately 50. This reduction stabilizes after 3 million timesteps. As training advances, fewer steps are needed for generating a complete episode, leading to improved efficiency. The bottom left graph shows a linear relationship between the cumulative time required for each step in the training process. The bottom right graph demonstrates a decreasing training loss, approaching zero as training progresses. This suggests that the model quickly converges after the first 20,000 steps.

- **Chapter 1: The Evil Queen**
- **Chapter 2: The Escape**
- **Chapter 3: The Huntsman's Failed Mission**
- **Chapter 4: The Poisoned Apple**
- **Chapter 5: True Love's Kiss**
  - **Chapter 5.2: The Arrival**
    - **Chapter 5.2.1: The Prince's Arrival**
      - **Chapter 5.2.1.1: Warm Welcome**
      - **Chapter 5.2.1.2: The Prince's Purpose**
      - **Chapter 5.2.1.3: The Dwarves' Praise**
  - **Chapter 5.3: True Love's Kiss**
  - **Chapter 5.4: Defeating the Evil Queen**

The story starts with the introduction of the villain, the Evil Queen. She is obsessed with her own beauty and jealous of anyone who is more beautiful than her. She learns about Snow White who is the fairest of them all and decides to get rid of her. **The Evil Queen's obsession with beauty had started at a young age.** As a child, she had always been told how pretty she was, and it soon became the defining aspect of her identity. Her parents would enter her into beauty pageants and modelling competitions, and she would always come out on top. However, as she grew older, she found it harder and harder to maintain her looks. Every wrinkle, every grey hair was a reminder of her mortality, and it terrified her. So, she indulged in every beauty treatment known to humankind, regularly ingested potions and pills, and even had some of her internal organs removed, all in the name of staying young and beautiful

...

...

...

Snow White had emerged victorious, and when the queen was finally captured, the people rejoiced. The kingdom had been restored to its former glory, and Snow White had proven herself to be an excellent ruler. The prince was proud of his partner and loved her more than ever. They had come a long way, and their shared experience had brought them closer together. Snow White was crowned queen of the kingdom, while the prince became her king. Together, they ruled with fairness and love, and the kingdom flourished under their wise leadership.

Fig. 3. Our method constructs a story by first determining each chapter's title and main content, then breaking it into subsections for coherence. Rather than deciding the total number of chapters upfront, it completes one chapter before starting the next.

This distinctive feature ensures a more comprehensive and satisfying storytelling experience for the readers.

According to the results described in Figure 2, the reward reaches a stable state at approximately $3e^6$ timesteps. Similarly, the episode length stabilizes around 50 after the same number of timesteps. The training duration exhibits a consistent increase, indicating a steady interaction with the simulator. Additionally, the training loss significantly decreases within the initial 20,000 steps. During the testing phase, we utilized ten different story titles, generated by GPT-3.5, as test samples. The results demonstrate that the final reinforcement learning model surpasses the initial model in terms of length, coherence, and consistency. [3]

Case Study As the text is more than 20,000 words, we select to show part of the chapters here as our case study in Figure 3. The story titled "Snow White" serves as the basis for this novel. The narrative has five chapters with sub-chapters, starting with the "Evil Queen" and ending with "Snow White's victory". While based on a known fairy tale, our version changes some details for a distinct story. Compared to initial

[3]More test results can be seen from https://bit.ly/3tYrYEx

TABLE II

UNLIKE RECURRENTGPT THAT USES INTERACTIVE STORYTELLING, AUTO-PROMPTED RL GENERATES LONGER, COHERENT STORIES IN ONE CLICK WITHOUT REPEATED HUMAN INTERACTION. IT RESULTS IN A MORE COMPLETE STORY STRUCTURE WITH CLEARER ENDINGS, AND MAINTAINS A SPECIFIC STORYLINE THROUGHOUT THE TEXT. CONVERSELY, RECURRENTGPT RELIES ON HUMAN-PROVIDED CUES AT EACH INTERACTION.

Features	RecurrentGPT	Auto-Prompted RL
Text Length	5000 words max (guaranteed content coherence and human evaluation)	20,000 words or more (guaranteed coherence and consistency)
Structural Closure	divergent with no real end	have a clear end of the article to keep the content complete
Story Plot	no specific plot, rely on the summary to control the conclusion	specific plot libraries control full-text generation

training, this story has a complex structure due to multiple tiers of chapters, adding vividness to the narrative. The inclusion of numerous events also adds variation and depth.

VI. CONCLUSION

In conclusion, effective long text generation via Pretrained Language Models (PLMs) depends on three considerations: generating and refining storylines, optimization for story plots text generation, and a systematic evaluation of long text requirements. Our proposed auto-prompted RL approach using GPT-3.5 notably addresses these considerations, as demonstrated by our experimental results. Through the initial creation of an outline followed by the expansion of each chapter using an auto-prompt system, we can effectively steer the model in a coherent direction. Simultaneously, we update the subsequent storyline. This not only ensures alignment with the given information and the narrative structure but also significantly enhances the overall text quality and consistency of the outlines. Our method demonstrates significant potential for the application of RL in various text generation tasks, including academic paper writing.

REFERENCES

[1] H. Zhang, H. Song, S. Li, M. Zhou, and D. Song, "A survey of controllable text generation using transformer-based pre-trained language models," *arXiv preprint arXiv:2201.05337*, 2022.

[2] R. Tang, Y.-N. Chuang, and X. Hu, "The science of detecting llm-generated texts," *arXiv preprint arXiv:2303.07205*, 2023.

[3] A. Alabdulkarim, S. Li, and X. Peng, "Automatic story generation: Challenges and attempts," *arXiv preprint arXiv:2102.12634*, 2021.

[4] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[5] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[6] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," *arXiv preprint arXiv:1904.09675*, 2019.

[7] X. Hu, Q. Chen, H. Wang, X. Xia, D. Lo, and T. Zimmermann, "Correlating automated and human evaluation of code documentation generation quality," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 4, pp. 1–28, 2022.

[8] J. D. Chang, K. Brantley, R. Ramamurthy, D. Misra, and W. Sun, "Learning to generate better than your llm," *arXiv preprint arXiv:2306.11816*, 2023.

[9] J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel, "A survey of reinforcement learning informed by natural language," *arXiv preprint arXiv:1906.03926*, 2019.

[10] N. McIntyre and M. Lapata, "Plot induction and evolutionary search for story generation," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 1562–1572.

[11] M. Eger, C. Potts, C. Barot, and R. M. Young, "Plotter: operationalizing the master book of all plots," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 11, no. 4, 2015, pp. 30–33.

[12] Z. Li, X. Ding, and T. Liu, "Generating reasonable and diversified story ending using sequence to sequence model with adversarial training," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 1033–1043.

[13] P. Jain, P. Agrawal, A. Mishra, M. Sukhwani, A. Laha, and K. Sankaranarayanan, "Story generation from sequence of independent short descriptions," *arXiv preprint arXiv:1707.05501*, 2017.

[14] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[15] S. Goldfarb-Tarrant, T. Chakrabarty, R. Weischedel, and N. Peng, "Content planning for neural story generation with aristotelian rescoring," *arXiv preprint arXiv:2009.09870*, 2020.

[16] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[17] T. Pradyumna, D. Murtaza, J. M. Lara, A. Mehta, and B. Harrison, "Controllable neural story plot generation via reward shaping," in *Proc. Int. Joint Conf. Artificial Intelligence*, 2019, pp. 5982–5988.

[18] S. Zhang, C. Gong, L. Wu, X. Liu, and M. Zhou, "Automl-gpt: Automatic machine learning with gpt," 2023.

[19] H. Dang, S. Goller, F. Lehmann, and D. Buschek, "Choice over control: How users write with large language models using diegetic and non-diegetic prompting," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–17.

[20] W. Zhou, Y. E. Jiang, P. Cui, T. Wang, Z. Xiao, Y. Hou, R. Cotterell, and M. Sachan, "Recurrentgpt: Interactive generation of (arbitrarily) long text," 2023.

[21] S. A. Crossley, "Linguistic features in writing quality and development: An overview," *Journal of Writing Research*, vol. 11, no. 3, pp. 415–443, 2020.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[23] C. Wang, Z. Guo, J. Li, G. Li, and P. Pan, "A text-based deep reinforcement learning framework using self-supervised graph representation for interactive recommendation," *ACM/IMS Transactions on Data Science (TDS)*, vol. 2, no. 4, 2022.

[24] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[25] A. Fan, M. Lewis, and Y. Dauphin, "Strategies for structuring story generation," *arXiv preprint arXiv:1902.01109*, 2019.

[26] P. Xu, M. Patwary, M. Shoeybi, R. Puri, P. Fung, A. Anandkumar, and B. Catanzaro, "Megatron-cntrl: Controllable story generation with external knowledge using large-scale language models," *arXiv preprint arXiv:2010.00840*, 2020.

[27] N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," 2023.

[28] S. Wang, G. Durrett, and K. Erk, "Narrative interpolation for generating and understanding stories," *arXiv preprint arXiv:2008.07466*, 2020.