Multi-Agent Based Character Simulation for Story Writing

Tian Yu [*]**, Ken Shi** [*]**, Zixin Zhao and Gerald Penn**
Department of Computer Science
University of Toronto
CANADA
{tianyu99,kenshi,zzhao1,gpenn}@cs.toronto.edu

Abstract

This work proposes a novel multi-agent story-generation system that writes stories from a narrative plan. Traditional approaches tend to generate a section of text directly from its outline. Our system, by contrast, divides this elaboration process into role-play and rewrite steps, where the former step enacts the story in chronological order with LLM-backed character agents, and the latter step refines the role-play result to align with a narrative plan. We show that the stories produced by our system are preferable to two other LLM-based story-generation approaches. We attribute this advancement to the benefits of incorporating a character-based simulation strategy.

1 Introduction

Recent advancements in Large Language Models (LLMs) have significantly improved text coherence and fluency. Researchers are now implementing automatic story generation and human-AI writing tasks using LLMs (Lee et al., 2024; Alabdulkarim et al., 2021). Traditionally, story generation involved a planning stage to sequence events, followed by a generation stage to elaborate these events into scenes (Alhussain and Azmi, 2021a). Approaches like IPOCL (Riedl and Young, 2010) treat narrative planning as a search problem using character or author goals as guides (Dehn, 1981; Meehan, 2013). Recent work proposes using LLMs as planning engines, such as Agents' Room (Huot et al., 2024), which uses multi-agent collaboration for narrative planning, and Dramatron (Mirowski et al., 2023), which modularizes the generation process in a manner similar to screenplay writing. Research shows that writers prefer modularizing story generation into smaller components, as it allows control over which parts of the process are automated (Lee et al., 2024; Reza et al., 2024; Mirowski et al., 2023).

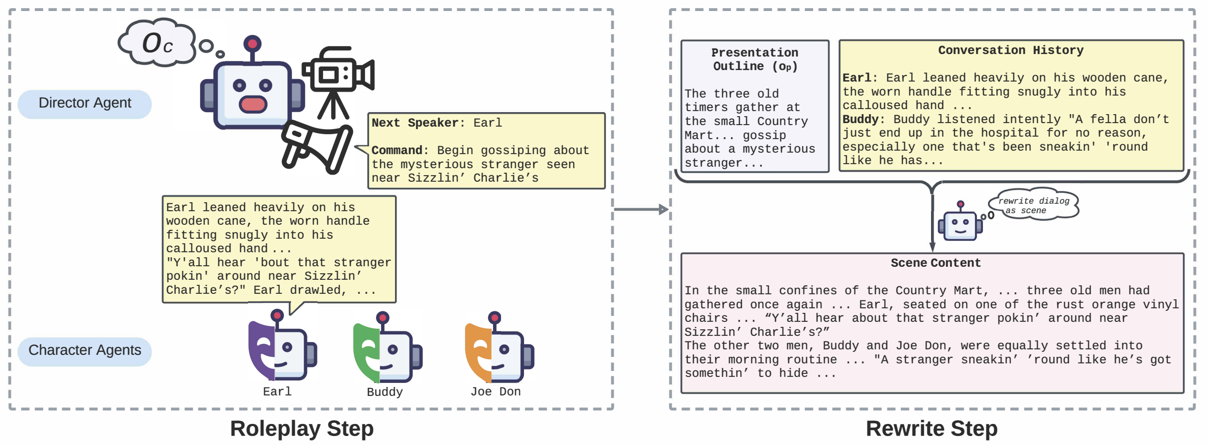
Despite the advances in automatic story generation, there remain many problems with LLM-generated stories. One is a lack of interest due to their linear nature (Alabdulkarim et al., 2021). Within narrative theory, stories can be separated into chronological time (*fabula*) and story time (*syuzhet*), where in many cases, stories can become more interesting when told in non-linear time (Liveley, 2019). There is, however, a lack of work in automatic story generation that looks at non-linear story generation, partially due to the risk of introducing inconsistencies and plot holes. Therefore, in our work, we propose generating a story through its fabula before re-organizing it into its syuzhet.

Our system aims to develop tools that integrate seamlessly into writers' workflows. We draw inspiration from hierarchical scriptwriting techniques (Mirowski et al., 2023) and adapt them for narrative story generation. By modularizing the writing process, we enhance control and facilitate human-LLM collaboration. Our method involves two steps (Figure 1): role play and rewrite. In the role play step, agents simulate scenes by acting as characters. In the rewrite step, the generated content is refined into story text. In the case of human-ai collaborative writing, during the role play step, the writers can act as an independent agent and role-play a character alongside LLM agents to simulate the scene. In the rewrite step, the writers can actively edit the intermediate scene content or use the generated content as inspiration to write their story.

Our work proposes a novel way to integrate the concepts of fabula and syuzhet into a cohesive story-generation process. Additionally, we present a novel multi-agent role-play approach by introducing a rewrite step. To evaluate our method, we compare stories generated by our system with two baseline methods using LLM-based automated evaluation, showing improvements in all aspects of story quality.

[*] Equal contribution

Proceedings of the Fourth Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2025), pages 87–108
May 4, 2025 ©2025 Association for Computational Linguistics



Next Speaker: Earl

Command: Begin gossiping about the mysterious stranger seen near Sizzlin' Charlie's

Earl leaned heavily on his wooden cane, the worn handle fitting snugly into his calloused hand ...
"Y'all hear 'bout that stranger pokin' around near Sizzlin' Charlie's?" Earl drawled, ...

The three old timers gather at the small Country Mart... gossip about a mysterious stranger...

Earl: Earl leaned heavily on his wooden cane, the worn handle fitting snugly into his calloused hand ...
Buddy: Buddy listened intently "A fella don't just end up in the hospital for no reason, especially one that's been sneakin' 'round like he has...

In the small confines of the Country Mart, ... three old men had gathered once again ... Earl, seated on one of the rust orange vinyl chairs ... "Y'all hear about that stranger pokin' around near Sizzlin' Charlie's?"
The other two men, Buddy and Joe Don, were equally settled into their morning routine ... "A stranger sneakin' 'round like he's got somethin' to hide ...

Figure 1: Overview of our character-simulation-based story-writing system, with details omitted. We break down the task of writing a story from a plan into two steps: role-play and rewrite. In the role-play step, the Director Agent will select and command Character Agents to respond, following the chronological outline of a scene (o_c). In the rewrite step, we prompt LLMs to write the actual content of the scene using the role-play results, along with the presentation outline (o_p) of that scene. The example is from train-000 in the Tell Me A Story dataset. Refer to Section 3.2 for details.

2 Related Work

2.1 Simulating Characters with LLMs

Characters are essential to many narrative stories as they often drive the plot and bring the narrative to life (LaPlante, 2007). Creating interesting and believable characters using LLMs has been explored by many prior papers (Li et al., 2023; Pichlmair et al., 2024; Wang et al., 2024b; Shao et al., 2023; Magee et al., 2024). Some previous work has focused on simulating believable character behaviour by introducing different aspects within character prompts such as "ego/superego" roles to simulate internal conflict (Magee et al., 2024) or behaviour trajectories using psychological grounding (Wang et al., 2024a). Other work has proposed using personality traits, routines, emotions, and social interactions (Zhao et al., 2024; Wang et al., 2024b; Yang et al., 2024; Normoyle et al., 2024). Overall, this work has found that LLMs can simulate nuanced and believable characters. Moreover, past work, such as Park et al. (Park et al., 2023), has also shown the viability of using LLMs to create generative agents that can produce believable individual and emergent social behaviours with memories of past interactions. Therefore, in our work, we hope to use LLMs to create believable characters within narrative stories.

2.2 Automatic Story Generation

Interesting characters alone do not make compelling stories; how they are revealed and evolve through events turns conversations into narratives. Early story generation models focused on plot using structural models from narrative theories like Propp's functions (Propp, 1968), or planning-based models guided by predefined goals, e.g., author goals (Dehn, 1981) or character goals (Meehan, 2013). These models often used planning agents to guide narrative generation, combining author and character goals to inform autonomous agents (Si et al., 2005). Social interactions among virtual agents can drive narrative diversity and emergent storytelling, avoiding rigid plot structures (Teutenberg and Porteous, 2013; Figueiredo et al., 2008; Porteous and Lindsay, 2019). Multi-agent approaches, as elsewhere, are preferred for their adaptability and control compared to monolithic systems.

Recent approaches use Seq2Seq models or LLMs to generate coherent stories from start to finish (Alhussain and Azmi, 2021a). Early Seq2Seq models struggled with coherence and consistency, but LLMs improved this, though long stories still lose cohesion due to context-window limits. To solve this, past work has proposed using multiple LLMs collaboratively (Venkatraman et al., 2024) or hierarchical story generation, separating plot planning from text generation (Fan et al., 2018; Yao et al., 2019). Dramatron (Mirowski et al., 2023) exemplifies this by modularizing story generation for screenplays, defining components like loglines (i.e. story premise), characters, plot, and locations, then generating dialogue. Furthermore, Dramatron was evaluated by professional screenplay writers who found that modularization allowed them to take over components of the generation and more control over the results.

Combining agent-based approaches with hierarchical story generation allows for user-controllable goals within LLM-based generation. For example, IBSEN (Han et al., 2024) uses a director-actor framework for script generation, while de Lima et al. (2022) combine multi-agent planning for interactive storytelling. DramaEngine(Pichlmair et al., 2024) and Agents' Room (Huot et al., 2024) use multi-agent workflows for narrative generation. Despite the promise of agent-based approaches, much work has focused on screenplays. Our work adapts hierarchical scriptwriting techniques (Mirowski et al., 2023; Han et al., 2024) to narrative story generation, modularizing the process to enhance control and facilitate human-LLM collaboration.

2.3 Automatic Story Evaluation

There are several well-established automatic text evaluation measures, such as perplexity (Brown et al., 1992), ROUGE (Lin, 2004), and BERTScore (Zhang* et al., 2020). These do not capture creativity, narrative structure, or coherence at the story level, however. They either reflect how 'typical' the text is or are reference-dependent, failing to measure aspects like creativity or storyline structure unless reference texts are carefully designed.

Recent work has attempted to evaluate stories with LLMs, often endorsing pairwise comparison. For example, Liusie et al. (2024) discussed using pairwise comparison for LLM evaluation. Subsequently, Liu et al. (2024b) found that pairwise comparison by LLMs aligns more closely with human evaluators than other methods. Additionally, Zheng et al. (2023) assessed the validity of using LLM evaluators through established benchmarks. In this work, we will also use LLM evaluators through pairwise comparison to assess the quality of generated stories, drawing inspiration from the LLM evaluator setup proposed by Agents' Room. It is worth noting that LLMs can suffer from the "Lost in the middle" effect (Liu et al., 2024a) when handling longer prompts. Therefore, we will implement specific measures to minimize this effect when using LLMs as evaluators.

3 Methodology

This section describes our approach to the outline-based creative writing task. We first provide an overview of our breakdown of the problem and then describe the individual components.

3.1 The Overall Task

In our work, we focus on the writing phase, where a narrative plan has already been provided for us to write the story. This approach is also used in other single and multi-agent frameworks. For example, Mirowski et al. (2023) used a hierarchical approach to generating screenplay dialogues based on a planned-scene outline, character information, and location details. Similarly, Huot et al. (2024) created a plan with multiple agents, each specializing in areas like character planning, and used the combined plan to write the story.

Our system draws inspiration from the classic narrative distinction between fabula and syuzhet (Alhussain and Azmi, 2021b). The fabula represents the raw, chronological sequence of events —– the underlying narrative as it unfolds in the story world. In our framework, the role-play step serves to develop this fabula, where character agents simulate the provided narrative plan in its chronological order.

Following this, the rewrite step takes the intermediate result and reshapes it into its final form, analogous to the syuzhet. This phase reorganizes and refines the content to align with the original plan. It optimizes the storytelling experience, much like how a narrative's presentation order can heighten dramatic effect and audience engagement.

3.2 Implementation Detail

3.2.1 The Definition and Agents

Our system takes in a plan and returns the actual story realized. We denote the **input plan** as P_p, which specifies the presentation order of scenes, representing the syuzhet. An input plan consists of a list of scenes to be written for the story. We define a **scene**, labelled S, as the minimum plan unit in our writing process. The scene includes a presentation outline (o_p) detailing a sequence of events (e), along with location information and the characters involved. To support character simulation, we define a character as an entity or a group of minor supporting entities in the story, with their name, gender, age, narrative role, setup, speaking characteristics, and character goal.

We define two types of agents: the director agent and the character agent. The **director agent**, labelled D, controls the scene's development in the role-playing process. It selects ($next\_speaker$) and instructs ($next\_command$) individual character agents. The **character agent**, labelled A, responds ($get\_response$) to the director's commands by considering its goal, physical state, and memory. These responses are described from a third-party perspective, providing realistic dialogue and action descriptions. Both agent types can be powered by either LLMs or human participants, enabling effective human-AI collaboration.

3.2.2 The Role-playing Step

While other LLM-based methods for narrative plan generation (Huot et al., 2024; Mirowski et al., 2023) rely on one LLM call to generate the content of a section, we divide the process into two subtasks, where the role-playing step is the first step.

Unlike RPG games, stories may not follow the chronological order to describe the events. As such, the input plan P_p provided by the users may contain two scenes S_i and S_j, where S_i happens after S_j chronologically, even though S_i should be presented earlier than S_j in the actual plan. As such, we define the **role-play plan** (P_c) consisting of the scenes from P_p in chronological order representing the fabula. A separate LLM-based sorting algorithm is used to create P_c from P_p. In this paper, we assume that for any two scenes $S_i, S_j \in S$, S_i cannot overlap S_j in the time domain, meaning any events in S_i will happen all before or all after the events in S_j. On top of this, another LLM-based sorting algorithm will generate a chronological outline (o_c) based on the presentation outline (o_p) of the scene and further refine it to be suitable for subsequence character-based role play and ensure that events within scenes are correctly ordered. Both LLM prompts are detailed in Appendix B.1.1, Prompt *Sort Scenes*, and Appendix B.1.2, Prompt *Chronological Outline Creation*, respectively.

After creating P_c and o_c, we obtain the full chronological development of the events in the whole story. Then, for each scene $S_i \in P_c$, we use algorithm 1 to role-play that scene. The other input, M, to the algorithm, is an accumulated map with the key being the name of the character agent and the value being the agent instance. We will re-use the same character agent, if it has already been created, to update the accumulated memory and, whenever it is involved, the physical state.

The role-playing logic for each scene is the same, where a group chat manager will be initialized for two tasks:

1. Determining whether the role-playing game has covered o_c, and terminate if so.
2. Guiding the role-playing game by following o_c, selecting the proper character agent to speak to, and providing them with the command of action.

When a character agent is selected, it first updates its internal states, including memory and physical state. In our implementation, we use a text-based memory and physical-state system, which updates based on the new chat history that the agent has not seen before. The agent then responds based on the role-playing game's history for consistency, the Director Agent's command, its memory and physical state, and its scene-level goal. This approach balances the *agent/character goal* (Riedl and Young, 2010), provided within the plan, with the *author goal* (Riedl, 2009), represented by the director agent's command. We instruct agents to respond in a third-person perspective to create realistic character dialogue and action descriptions, catering to the story-generation use case. Please refer to Appendix B.1 for the relevant prompts used.

3.2.3 The Rewrite Step

The role-playing output may not produce a perfect story, however, because it only replicates the fabula as part of the story world. A story should be produced by viewing the fabula from a specific angle (Swartjes and Theune, 2006), following P_p, the presentation order of the initially planned scenes. As mentioned in Section 3.2.2, our role-playing result follows o_c for the sequence of events in a scene and P_c for the sequence of scenes. This means $o_c \neq o_p$ and $P_c \neq P_p$ are possible.

To address this, we implement a re-writing algorithm. For each scene, we prompt LLMs to write the scene content based on the presentation outline o_p, referencing the corresponding simulation results in the role-play step for character dialogues and actions. We also generate the scene content sequentially, following the presentation order specified in the input plan P_p. This modular approach allows authors to revise the content of each scene before it is utilized in the subsequent scene generation. The prompts used for this step are available in Appendix B.2.

Algorithm 1 Scene-level role-playing.

```
Require: Scene $S_i$,
         Character Agents Map $M$
Init Chat History, $H \leftarrow [\,]$                          ▷ $H$ will store all messages in the session
Init Director Agent $D.init()$
while not $D$.should_terminate($S_i, H$) do                   ▷ Check if the scene should end
    $A_i \leftarrow D$.next_speaker($S_i, H$)                   ▷ The director selects the next character agent
    if $A_i.name \in M$ then
        $A_i \leftarrow M.get(A_i.name)$                        ▷ If this agent exists, retrieve the agent
    else
        $M$.add($A_i.name, A_i$)                                ▷ If this agent is new, add it to the map
    end if
    $C_j \leftarrow D$.next_command($S_i, H$)                   ▷ The director selects the next command
    $A_i$.update_state($H$)                                     ▷ The chosen agent updates its internal memory
    $h_i \leftarrow A_i$.get_response($H, C_j$)                 ▷ The agent generates a response
    $H$.append($h_i$)                                           ▷ Add the agent's response to the chat history
end while
return $H$                                                      ▷ Return the full history once the scene is complete
```

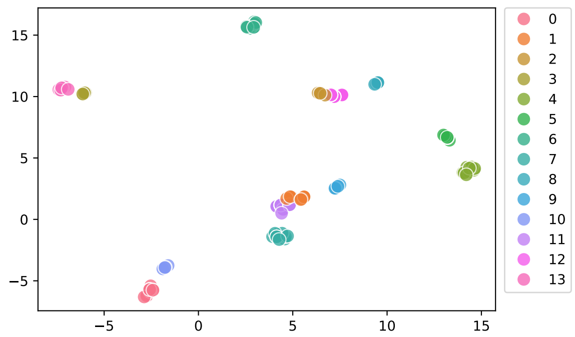


Figure 2: Writing prompts from the **Tell Me A Story** dataset, clustered into 14 groups using UMAP and k-means.

4 Experiments

4.1 Dataset

For our dataset, we used **Tell Me A Story**,[1] made up of complex writing prompts and human-written stories. This dataset is contains 230 prompts in total, but upon manual inspection, we found that many of the prompts differed by only a few words. Therefore, we first evaluated the number of unique prompts present in the dataset. To do this, we first created sentence embeddings using sBERT (Reimers and Gurevych, 2019), then reduced the dimensions of the embedding using UMAP (McInnes et al., 2018) before using k-means to assign each sentence to a cluster. We tested various numbers of clusters and found that the number that fit the data best was 14, shown in Figure 2. After determining the number of writing prompts clusters, we manually looked through the data and selected 28 representative prompts so that we would have coverage over the range of stories that could be generated with this dataset.

4.2 Experiment Setup

In this section, we describe our setup for the experiment to prove the validity of our method. We compare our approach with two approaches:

1. The single-agent-based approach, where Dramatron (Mirowski et al., 2023) is the baseline,

2. The multi-agent-based approach, where Agents' Room (Huot et al., 2024) is the baseline.

At a high level, as shown in Figure 3, our experiment can be treated as a back-translation process between the gold story and the synthetic plan, where we first generate the plan using a teacher LLM model and then use each system to write the final story, given the plan created.

4.2.1 Plan Synthesis

We can treat each of Dramatron's and Agents' Room's story-writing approaches as a sequence of planning and writing tasks. By the writing portion of Dramatron, we intend to refer to the dialogue

[1] https://github.com/google-deepmind/tell_me_a_story

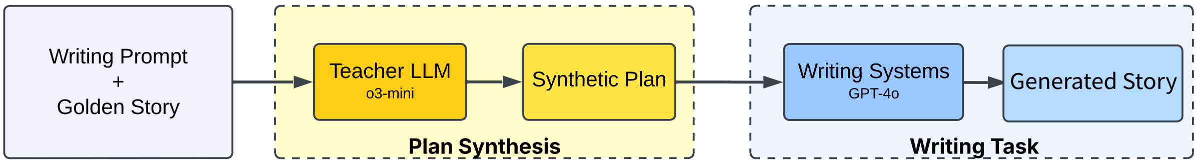


Figure 3: Experimental overview: given the writing prompt and the gold story, we first generate a synthetic plan using a Teacher LLM. The plan is then used to generate the final story.

generation step that follows the hierarchical generation process. We consider the writing portion of the Agents' Room approach following what is mentioned in their paper. Since a plan is required for all three systems, including ours, we applied a similar approach as in previous work ((Huot et al., 2024; Schick et al., 2022; Josifoski et al., 2023), where the expected plan is synthesized to generate through distilled back-translation.

For the Agents' Room implementation, we provide the writing prompt and gold story to a teacher LLM (O3-mini) to obtain the story's central conflict, characters, setting and plot. We followed exactly the prompt used in their work to extract the plan.

For our system, we first prompt the teacher LLM (O3-mini) to generate all the characters that appeared in the gold story, providing both the writing prompt and the gold story. We then prompt the teacher LLM (O3-mini) with the additional character extracted to obtain the list of scenes that happened in the gold story, denoted by P_p in Section 3.2.2. In addition, we also consider the central conflict and setting generated using the same method as the Agents' Room part of the plan.

For Dramatron, we share the plan information with the one generated for our system, as no details are provided in their work. Specifically, we provide the place and character information, along with the sequence o_p from the P_p extracted.

4.2.2 Writing Task

We defined similar writing tasks for all methods tested. The LLM used for our experiments was the GPT-4o model, with the temperature set to 0.9, and frequency penalty equal to 0.2. We used a zero-shot prompting strategy for all systems for a fair comparison. Here, we go into the details of each method: Agents' Room (Huot et al., 2024), Dramatron (Mirowski et al., 2023), and our method.

First, for the Agents' Room writing task, we followed the procedures mentioned in their work, where five agents are created with the prompt provided in their paper, each writing a stage of a narrative arc (exposition, rising action, climax, falling action and resolution). The final story is the five agents' output, concatenated sequentially.

For Dramatron, we had to modify their approach so that we could generate the story scenes rather than scenes made up of screenplay dialogues. Additionally, we removed their few-shot examples, which would be incomparable if included. Otherwise, we followed the exact implementation for the prompt as shared in their codebase[2] to the best of our ability and retained the scene-by-scene generation process. The changed prompts are shown in Appendix B. As for our method, we used the approach described in Section 3.2.

4.3 Evaluation Method

In this work, we take inspiration from the automatic evaluation used in Agents' Room (Huot et al., 2024) to build an LLM-based evaluator. We chose to use the same set of criteria that align with Agents' Room's LLM-based evaluation. Specifically, we construct prompts that will evaluate the story in terms of four criteria, namely, *Plot*, *Creativity*, *Development* and *Language Use*.

We provide a template of a prompt that specifies the above criteria as its aspects, along with a presentation of the pair of stories to be compared. The detailed definition of each criterion is written in the template, which can be found in Appendix B.5. To align with Agents' Room's choice of evaluator model, we also use Gemini 1.5 Pro.

As mentioned earlier, to ensure the fairness of the comparison, we evaluate each pair of stories twice by swapping the presentation order of the stories within the evaluation prompt.

For each of the 5 criteria (*overall* is a separate criterion, as we explicitly ask the evaluator to generate an“overall” decision), c, we perform evaluations on all pairs of stories that are of the same writing prompt across all pairs of systems to ob-

[2] https://github.com/google-deepmind/dramatron

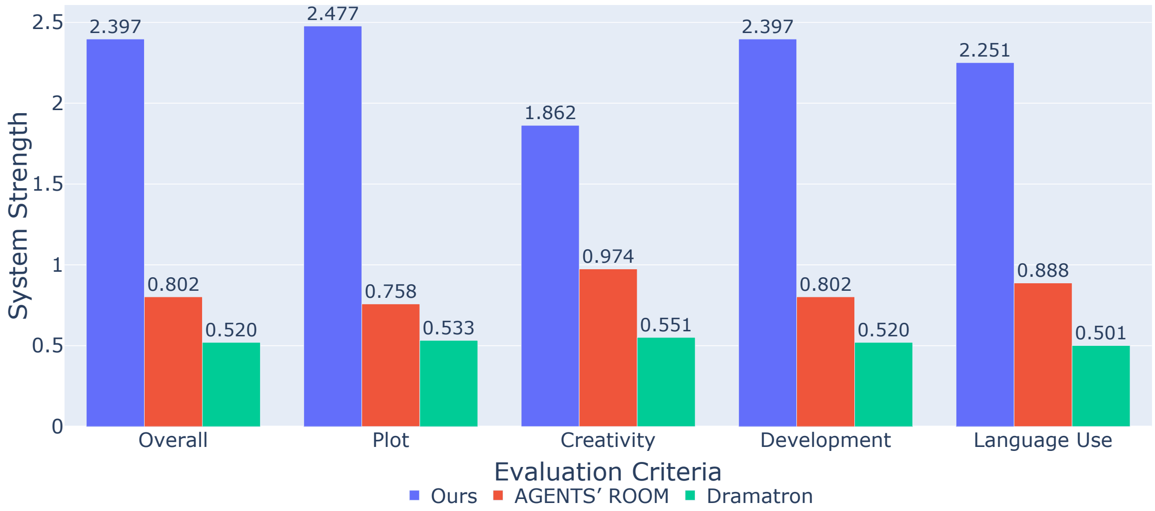


Figure 4: System strength across dimensions of plot, creativity, development, language use and overall, according to the LLM-based evaluator.

tain a win matrix $W^c \in \mathbf{R}^{N \times N}$, where N is the number of systems. Each element $W^c_{i,j}$ represents the number of times system i beats system j. We then linearize these pairwise comparisons using the Bradley-Terry model to obtain latent ability parameters, which denote the numerically ordered strengths of each system with respect to c. To present the result, we follow Agents' Room's convention of normalizing the log strengths, centered around 0.

5 Results

5.1 LLM Evaluation Results

Figure 4 demonstrates the strength of the three systems across the set of criteria defined at the beginning of Section 4.3. Overall, our system outperforms both Agents' Room and Dramatron. This behaviour is consistent across each criterion. The values of each win matrix can be found in Appendix A.

In terms of strength, our system comes out on top among all the criteria, and by a large gap. These are easily misinterpreted, however. BTL strengths are not a measure of how *much* better one system's stories are over another's, but rather of how *often* the one's are preferred (here, by the LLM evaluator) over the other's. This method can be interpreted by analogy to a consumer-product trial, in which the LLM evaluator samples a population of consumers, and the generated stories are the products being tried. The result of the trials only indicates with what likelihood a consumer might choose one brand over another.

The results are generally consistent when swapping the order of presentation for each pair of stories. In cases where inconsistency is observed, it often occurs in stories where the various criteria conflict. Such inconsistencies, however, are handled by the nature of our design, which treats them as less rewarding than a consistent win and more rewarding than a consistent loss.

5.2 Qualitative Analysis

In addition to the LLM evaluation above, we investigated the individual stories generated by those systems. One observation we made was that the stories generated by our system often maintain better character consistency and narrative coherence. For example, train 026 golden story primarily revolves around the interactions between Scholar Kissen and Courier Aerie. The first scene portrays their initial meeting, whereas the second involves Aerie recalling her earlier journey to a remote site, illustrating a non-chronological scene arrangement. As such, the extracted plan from the story poses a significant challenge for the Dramatron system to create the narration of the first scene, given that the model inherently lacks awareness of chronologically earlier events that have nevertheless not yet been narrated (as with the second scene). This results in an hallucination of Aereo's dialogue about the site being created. Conversely, systems such as Agents' Room, which provide a full story outline upfront, face the risk of prematurely revealing information. Specifically, in the first scene, Aerie's dialogue preemptively references details that should appear later in the story, thus disrupting the narrative flow. Our approach is more balanced. By employing the sorting mechanism to role-play chronologically, the character agent, Aerie, has the memory of her visit to the site (second scene) before role-playing the first scene (meeting scholar Kissen). Additionally, the rewriting mechanism ensures that the generated scene content strictly adheres to the current scene outline, effectively preventing the premature disclosure of future information and resulting in a more consistent and coherent story. Please refer to Appendix C.1 for details.

It is also observed that the other approaches are not as consistent in producing high-quality stories in the long run. For the Agents' Room approach, we have spotted repetition and, occasionally, random off-topic words generated. We believe this is caused by the generation process being too weakly constrained, in which only a few constraints other than the plot line are provided to guide the generation.

Another observation of our proposed system is that sometimes the group chat manager repeats the same command to the character agents when it is unsatisfied with the agent's response. This often happens to content corresponding to plots near the end of the scene outline. To mitigate this issue, we set a maximum number of 10 iterations, which avoids the potential for infinite repetition.

6 Conclusion

In conclusion, our work is the first to integrate the concepts of fabula and syuzhet into a unified process for generating stories from a narrative plan. We decomposed story creation into two distinct phases/steps: a fabula generation phase (role-play step) driven by realistic, LLM-backed character agents under the guidance of a director agent. This achieves a natural balance between authorial intent and character-driven conversation history. This is followed by a syuzhet modification phase (rewrite step), which refers to the conversation history and can potentially reuse the majority of dialogues and actions, only needing to manipulate their order, thus significantly reducing the difficulty of the actual story-content-writing process.

Our current approach assumes that a scene consists of a sequence of events occurring within a specific location and time frame. However, there are cases where scenes interleave between present conversations and flashbacks — for example when a character recalls a memory. As such, our sorting algorithm may fail to produce a strictly chronologically ordered plan for the role-play step. Future work could address this limitation by sorting all events in the story with finer granularity, regardless of scene boundaries, to handle complex temporal structures better. Additionally, while our system updates agents' memories to reflect only the information they should see, it does not explicitly enforce privacy during the role-play process. Future work can improve upon this by implementing explicit privacy controls. Lastly, our LLM evaluation method aligns with human-tested criteria. Since our project's goal is to assess the potential of agent-based simulation for story creation, future work can explore effective approaches for integrating human participants into this process.

Acknowledgments

This research was supported by a grant from NAVER corporation. Besides, we thank our anonymous reviewers for their very helpful suggestions.

References

Amal Alabdulkarim, Siyan Li, and Xiangyu Peng. 2021. Automatic story generation: Challenges and attempts. In *Proceedings of the Third Workshop on Narrative Understanding*, pages 72–83, Virtual. Association for Computational Linguistics.

Arwa I. Alhussain and Aqil M. Azmi. 2021a. Automatic story generation: A survey of approaches. *ACM Comput. Surv.*, 54(5).

Arwa I. Alhussain and Aqil M. Azmi. 2021b. Automatic story generation: A survey of approaches. *ACM Comput. Surv.*, 54(5).

Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, and Robert L Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Edirlei Soares de Lima, Bruno Feijó, and Antonio L. Furtado. 2022. A character-based model for interactive storytelling in games. In *2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 1–6.

Natlie Dehn. 1981. Story generation after tale-spin. In *IJCAI*, volume 81, pages 16–18.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Rui Figueiredo, António Brisson, Ruth Aylett, and Ana Paiva. 2008. Emergent stories facilitated an architecture to generate stories using intelligent synthetic characters.

Senyu Han, Lu Chen, Li-Min Lin, Zhengshan Xu, and Kai Yu. 2024. IBSEN: Director-actor agent collaboration for controllable and interactive drama script generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1607–1619, Bangkok, Thailand. Association for Computational Linguistics.

Fantine Huot, Reinald Kim Amplayo, Jennimaria Palomaki, Alice Shoshana Jakobovits, Elizabeth Clark, and Mirella Lapata. 2024. Agents' room: Narrative generation through multi-step collaboration. *arXiv preprint arXiv:2410.02603*.

Martin Josifoski, Marija Sakota, Maxime Peyrard, and Robert West. 2023. Exploiting asymmetry for synthetic training data generation: Synthie and the case of information extraction. *arXiv preprint arXiv:2303.04132*.

Alice LaPlante. 2007. *The Making of a Story: A Norton Guide to Creative Writing*. W. W. Norton & Company, New York, NY.

Mina Lee, Katy Ilonka Gero, John Joon Young Chung, Simon Buckingham Shum, Vipul Raheja, Hua Shen, Subhashini Venugopalan, Thiemo Wambsganss, David Zhou, Emad A Alghamdi, and 1 others. 2024. A design space for intelligent and interactive writing assistants. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–35.

Cheng Li, Ziang Leng, Chenxi Yan, Junyi Shen, Hao Wang, Weishi Mi, Yaying Fei, Xiaoyang Feng, Song Yan, HaoSheng Wang, and 1 others. 2023. Chatharuhi: Reviving anime character in reality via large language model. *arXiv preprint arXiv:2308.09597*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yinhong Liu, Han Zhou, Zhijiang Guo, Ehsan Shareghi, Ivan Vulić, Anna Korhonen, and Nigel Collier. 2024b. Aligning with human judgement: The role of pairwise preference in large language model evaluators. In *First Conference on Language Modeling*.

Adian Liusie, Potsawee Manakul, and Mark Gales. 2024. LLM comparative assessment: Zero-shot NLG evaluation through pairwise comparisons using large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 139–151, St. Julian's, Malta. Association for Computational Linguistics.

Genevieve Liveley. 2019. *Narratology*. Oxford University Press.

Liam Magee, Vanicka Arora, Gus Gollings, and Norma Lam-Saw. 2024. The drama machine: Simulating character development with llm agents. *arXiv preprint arXiv:2408.01725*.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

James Meehan. 2013. Tale-spin. In *Inside Computer Understanding*, pages 197–226. Psychology Press.

Piotr Mirowski, Kory W Mathewson, Jaylen Pittman, and Richard Evans. 2023. Co-writing screenplays and theatre scripts with language models: Evaluation by industry professionals. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–34.

Aline Normoyle, João Sedoc, and Funda Durupinar. 2024. Using llms to animate interactive story characters with emotions and personality. In *2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 632–635.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA. Association for Computing Machinery.

Martin Pichlmair, Riddhi Raj, and Charlene Putney. 2024. Drama : for. Technical, Write with LAIKA, Copenhagen, Denmark.

Julie Porteous and A. Lindsay. 2019. Protagonist vs antagonist provant: Narrative generation as counter planning. In *Adaptive Agents and Multi-Agent Systems*.

Vladimir Propp. 1968. *Morphology of the Folktale*. University of Texas Press.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Mohi Reza, Nathan M Laundry, Ilya Musabirov, Peter Dushniku, Zhi Yuan “Michael” Yu, Kashish Mittal, Tovi Grossman, Michael Liut, Anastasia Kuzminykh, and Joseph Jay Williams. 2024. Abscribe: Rapid exploration & organization of multiple writing variations in human-ai co-writing tasks using large language models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–18.

Mark O Riedl. 2009. Incorporating authorial intent into generative narrative systems. In *AAAI Spring Symposium: Intelligent Narrative Technologies II*, pages 91–94.

Mark O Riedl and Robert Michael Young. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268.

Timo Schick, Jane Dwivedi-Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2022. Peer: A collaborative language model. *arXiv preprint arXiv:2208.11663*.

Yunfan Shao, Linyang Li, Junqi Dai, and Xipeng Qiu. 2023. Character-LLM: A trainable agent for role-playing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13153–13187, Singapore. Association for Computational Linguistics.

Mei Si, Stacy C. Marsella, and David V. Pynadath. 2005. Thespian: using multi-agent fitting to craft interactive drama. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’05, page 21–28, New York, NY, USA. Association for Computing Machinery.

Ivo Swartjes and Mariët Theune. 2006. A fabula model for emergent narrative. In *International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, pages 49–60. Springer.

Jonathan Teutenberg and Julie Porteous. 2013. Efficient intent-based narrative generation using multiple planning agents. In *Adaptive Agents and Multi-Agent Systems*.

Saranya Venkatraman, Nafis Irtiza Tripto, and Dongwon Lee. 2024. Collabstory: Multi-llm collaborative story generation and authorship analysis. *arXiv preprint arXiv:2406.12665*.

Lei Wang, Jianxun Lian, Yi Huang, Yanqi Dai, Haoxuan Li, Xu Chen, Xing Xie, and Ji-Rong Wen. 2024a. Characterbox: Evaluating the role-playing capabilities of llms in text-based virtual worlds. *arXiv preprint arXiv:2412.05631*.

Yi Wang, Qian Zhou, and David Ledo. 2024b. Storyverse: Towards co-authoring dynamic plot with llm-based character simulation via narrative planning. In *Proceedings of the 19th International Conference on the Foundations of Digital Games*, FDG ’24, New York, NY, USA. Association for Computing Machinery.

Bohao Yang, Dong Liu, Chenghao Xiao, Kun Zhao, Chen Tang, Chao Li, Lin Yuan, Guang Yang, Lanxiao Huang, and Chenghua Lin. 2024. Crafting customisable characters with llms: Introducing simschat, a persona-driven role-playing agent framework. *arXiv preprint arXiv:2406.17962*.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Runcong Zhao, Wenjia Zhang, Jiazheng Li, Lixing Zhu, Yanran Li, Yulan He, and Lin Gui. 2024. Narrative-Play: Interactive narrative understanding. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 82–93, St. Julians, Malta. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

A The Win Matrices of Section by Evaluation Criteria 5.1

$$\text{Overall} = \begin{pmatrix} 0 & 41 & 47 \\ 15 & 0 & 33 \\ 9 & 23 & 0 \end{pmatrix} \tag{1}$$

$$\text{Plot} = \begin{pmatrix} 0 & 42 & 47 \\ 14 & 0 & 32 \\ 9 & 24 & 0 \end{pmatrix} \tag{2}$$

$$\text{Creativity} = \begin{pmatrix} 0 & 36 & 44 \\ 20 & 0 & 35 \\ 12 & 21 & 0 \end{pmatrix} \tag{3}$$

$$\text{Development} = \begin{pmatrix} 0 & 41 & 47 \\ 15 & 0 & 33 \\ 9 & 23 & 0 \end{pmatrix} \tag{4}$$

$$\text{Language Use} = \begin{pmatrix} 0 & 39 & 47 \\ 17 & 0 & 34 \\ 9 & 21 & 0 \end{pmatrix} \tag{5}$$

B The Prompt Used

B.1 Our System - Role Play Step

B.1.1 Plan Synthesis and Sorting

Character Extraction

You are provided with:

1. <Writing Prompt>: This is the original instruction that was used to generate a story.
2. <Story>: This is the actual narrative generated based on the writing prompt.
3. <Characters>: A list of **most of the characters** that appeared in the story following the following schema: input_story_characters_schema

{{divider}}
Based on <Story>, and <Characters>, and also consider the <Writing Prompt>, divide the **ENTIRE** <Story> into scenes. A scene is a unit of story that takes place in a single location and time. Each scene should be a self-contained unit that moves the story forward, and try to divide the story based on the plot elements: exposition, rising action, climax, falling action, resolution.
For each divided scene content, considering the <Characters> information, you should provide the following information:

1. name: A high level name for the scene
2. outline: The outline of the scene, which is a description of the action/story/dramatic event occurring in the scene. It should comprehensively capture the actions and interactions of all characters involved. Answer in a list of bullet points
3. plot_element: Which plot element is being developed in this scene. You can choose from the following: exposition, rising action, climax, falling action, resolution.
4. place: The place of the scene with a SPECIFIC AND DETAILED description of that place
5. importance: The relative importance of the scene in the story. It is an integer in the scale 1-10, where 1 is the least important and 10 is the most important. You should provide importance based on the number of words of this scene compared to the total number of words in the story, and also the significance of the scene in the story.
6. characters: **ALL** the characters that **are present in the scene**. Also provide the scene level **CHARACTER GOAL** of each character. If you find any additional characters/group not provided in <Characters>, add them to the scene, and provide their character goal as well.

{{divider}}
Once you have divided the story into scenes, reflect on the scenes you have created, and ensure that all content is covered. If you have missed any content, add additional scenes/details to the outline to cover the missing content.
{{divider}}
Provide your output in JSON following the following schema:
{{OutScenesSchema}}

Scene Extraction

You are provided with:

1. <Writing Prompt>: This is the original instruction that was used to generate a story.

2. <Story>: This is the actual narrative generated based on the writing prompt.

3. <Characters>: A list of **most of the characters** that appeared in the story following the following schema: input_story_characters_schema

{{divider}}
Based on <Story>, and <Characters>, and also consider the <Writing Prompt>, divide the **ENTIRE** <Story> into scenes. A scene is a unit of story that takes place in a single location and time. Each scene should be a self-contained unit that moves the story forward, and try to divide the story based on the plot elements: exposition, rising action, climax, falling action, resolution.
For each divided scene content, considering the <Characters> information, you should provide the following information:

1. name: A high level name for the scene

2. outline: The outline of the scene, which is a description of the action/story/dramatic event occurring in the scene. It should comprehensively capture the actions and interactions of all characters involved. Answer in a list of bullet points

3. plot_element: Which plot element is being developed in this scene. You can choose from the following: exposition, rising action, climax, falling action, resolution.

4. place: The place of the scene with a SPECIFIC AND DETAILED description of that place

5. importance: The relative importance of the scene in the story. It is an integer in the scale 1-10, where 1 is the least important and 10 is the most important. You should provide importance based on the number of words of this scene compared to the total number of words in the story, and also the significance of the scene in the story.

6. characters: **ALL** the characters that **are present in the scene**. Also provide the scene level **CHARACTER GOAL** of each character. If you find any additional characters/group not provided in <Characters>, add them to the scene, and provide their character goal as well.

{{divider}}
Once you have divided the story into scenes, reflect on the scenes you have created, and ensure that all content is covered. If you have missed any content, add additional scenes/details to the outline to cover the missing content.
{{divider}}
Provide your output in JSON following the following schema:
{{OutScenesSchema}}

Sort Scenes

You are a creative writer for the story. Your task is to sort an array of story scenes based on the chronological order, and provide the sorted result.
{{divider}}
You are provided with:
1. <StoryScenes>: The list of story scenes, following the schema: {{input_story_scenes_schema}}
{{divider}}
Your Task is to
1. Sort the <StoryScenes> based on the chronological order of the story development by each scene's outline, and provide the sorted result, by the name of each scene.
{{divider}}
Output in JSON format following the schema provided below: {{sort_scene_results_schema}}

B.1.2 Director Agent

Chronological Outline Creation

You are provided with the following information:

1. <Scene>: The scene object, which includes both scene outline, detailed location description and involved character/group of characters information following the schema: {{input_scene_schema}}

2. <Outline>: The outline of the scene you are adjusting divider

Sort, and rewrite the scene outline bullet points to be suitable for a role-playing game (RPG). Ensure that:

- Strict chronological order: Events must be structured in the order they occur, avoiding retrospective narration (e.g., no "recounting" of past events).
- The outline focuses on character-driven development and role-playing dynamics
- The sequence of events reflects meaningful interactions between characters
- The updated outline should be similar in format. The events MUST BE in CHRONOLOGICAL ORDER, and described in present tense
- The updated outline should have similar number of word as the original <Outline> provided.
- Do not add any event, only reorder original events mentioned in <Outline> provided.

return the bullet points in str

Group Chat Termination

You are the director of a scene in a role playing game, and you are responsible for GUIDE the agents to act and speak according to the scene outline.
{{divider}}
You are provided with the following information:

1. <Scene>: The scene object, which includes both scene outline in chronological order for this RPG game, detailed location description and involved character/group of characters information following the schema: input_scene_schema

2. <ChatHistories>: The history of the role playing game, which is a sequence of message from participating characters in an array following the schema: {{input_chat_histories_schema}}

3. <NextAgentNames>: This is an array of str, representing the name of the characters that you are able to select to speak next. You should select **EXACTLY ONE NAME TO RESPOND**, and provide the **EXACT NAME OF THE AGENT**.

{{divider}}
First, review the <ChatHistories> provided, and also look at the <Scene> outline, and decide if the chat has covered the outline of the scene. And provide the reasoning of your decision. The reasoning must be specific, in terms of exact character and event in the scene outline not yet covered.
Then Answer True if the chat has covered the entire content, where the director should terminate the conversation, or False otherwise.
{{divider}}
Output in JSON format following the schema provided below: {{out_director_should_terminate_schema}}

Director Command and Select Agent

You are the director of a scene in a role playing game, and you are responsible for GUIDE the agents to act and speak according to the scene outline.
{{divider}}
You are provided with the following information:

1. <Scene>: The scene object, which includes both scene outline, detailed location description and involved character/group of characters information following the schema: input_scene_schema

2. <ChatHistories>: The history of the role playing game, which is a sequence of message from participating characters in an array following the schema: input_chat_histories_schema

3. <NextAgentNames>: This is an array of str, representing the name of the characters that you are able to select to speak next. You should select **EXACTLY ONE NAME TO RESPOND**, and provide the **EXACT NAME OF THE AGENT**.

{{divider}}
1. Continuation Planning:
- Examine the <ChatHistories>, which document the role-playing game progress so far. - Based on this conversation history, repeat **EXACTLY** the remaining part of the **outline** of the <Scene> provided that is not shown in the <ChatHistories>.
2. Agent Selection and Command:
- From your continuation plan, which provides the remaining scene outline to be role-played, choose which character agent should role-play next. Provide the exact name of that agent. - Directly address the chosen agent with a concise, high-level command for one turn. The command should provide a summary directive—indicating the intended action or dialogue direction—tailored to the character's age, gender, and personality. Avoid including detailed dialogue or overly specific descriptions. **Be concise**
{{divider}}
Generate the output in JSON following the following schema: {{out_director_selection_command_schema}}

B.1.3 Character Agent

Character Agent Response

You are acting as an agent in a role-playing game. You will produce responses on behalf of the agent from a third-person perspective, describing both the agent's actions and dialogue. Adhere to the agent's goals, age, gender, and personality at all times, **ensuring the response reflects their memory and physical state in a logical way.**
{{divider}}
You are provided with the following information:

1. <Character>: The character you are role playing for, and you should keep in mind the Character's goal, and act accordingly and realistically. It follows the schema: {{input_story_character_schema}}
2. <DirectorCommand>: The command from the director of the role play game, representing what the agent should incoporate to say and do in the role playing game.
3. <CharacterMemory>: The memory of the character, in string.
4. <CharacterPhysicalState>: The physical state of the character, in string.
5. <RecentHistories>: The most recent, up to 10 histories of the role playing game, following the schema: {{input_chat_histories_schema}}

{{divider}}
Based on <Character>, <DirectorCommand>, <CharacterMemory>, <CharacterPhysicalState>, and <RecentHistories>, generate the agent's response from a third-party perspective. The dialogue, actions, and overall tone must be **natural** realistic, taking into account the agent's age, background, personality, and speech patterns.
Important Guidelines: 1. Do not include any concluding commentary—only provide the agent's response in the role playing game. 2. Maintain an observer's perspective, presenting the agent's actions and dialogue authentically while ensuring alignment with their character traits. 3. Consider the agent's current memory and physical state, ensuring the response is realistic, concise, and free of contradictions with their established characteristics.
Generate the response in JSON in the following format:
{{out_response_schema}}

Character Agent Update Memory

You are the mechanism to update the current character's memory given the history of a role playing game.
{{divider}}
You are provided with:
1. <NewChatHistories>: The list of conversation and action history of the agents in the role playing game that is **not yet seen by the current character** following the schema: {{chat_histories_schema}}
2. <Character>: The character whose memory you are updating, following the schema:{{story_character_schema}}
3. <CharacterMemory>: The current memory of the character whose memory you are updating.
{{divider}}
Update the memory on what the current character should know based on the history, and return the updated memory. The memory should contain the history of events that the character has experienced, and any information that the character has learned from the conversation. Do not include any irrelavant information, and the memory should be in first character standpoint. For the output, only provide the **updated** memory in string, nothing else.

Physical State Update

You are the mechanism to record the physical state of the character based on the history of a role playing game.
{{divider}}
You are provided with:

1. <NewChatHistories>: The list of conversation and action history of the agents in the role playing game that is **not yet seen by the current character** following the schema: {{chat_histories_schema}}
2. <Character>: The character whose physical state you are updating, following the schema: {{story_character_schema}}
3. <CharacterPhysicalState>: The physical state of the character whose physical state you are updating.

{{divider}}
Update the physical state to reflect the changes based on the history. The physical state must be consistent with the <Character>, in terms of their age, gender and set up, and also make sense based on the the <NewChatHistories>. For the output, only provide the **updated** physical state in string, nothing else.

B.2 Our System - Rewrite Step

We share the same extraction process as AR in the creation of central conflict, story-setting. Please refer to their work for details.

Rewrite to Story

You are a creative writer writing a story. Your task is to write the content of the <TaskScene> for the story.
{{divider}}
You are provided with:

1. <WritingPrompt>: The writing prompt for the story
2. <CentralConflict>: The central conflict of the story
3. <StorySetting>: The setting of the story
4. <StoryScenes>: The list of story scenes planned out, by name and outline, following the schema: input_story_scenes_v2_schema
5. <StoryContent>: The story content written so far
6. <SceneCharacters>: The characters involved in the scene you are writing, following the schema: input_characters_schema
7. <TaskScene>: The scene object you are writing the content for, following the schema: input_story_scene_v2_schema
8. <TaskScenePlotElement>: The plot element of the scene you are writing
9. <TaskScenePlace>: The place where the scene unfolds.
10. <RolePlayHistory>: The conversation of <SceneCharacters> role playing <TaskScene> in an array, following the schema: input_chat_histories_v2_schema

{{divider}}
Besure to understand the <TaskScene>'s role in the whole narrative arc, and write the content of the scene accordingly.
Refer to the <RolePlayHistory> for **realistic character actions and dialogues** in an RPG game of the <TaskScene>. But begin your portion of the story in a way that naturally flows from the ending of <Story>. Match the writing style, vocabulary, and overall mood of the existing text. Do not re-explain details or events that have already been described. Ensure dialogue and actions **align with character traits**
{{no_end_instruction}}

1. <WritingPrompt>: {{writing_prompt}}
2. <CentralConflict>: {{central_conflict}}
3. <StorySetting>: {{story_setting}}
4. <StoryScenes>: {{story_scenes}}
5. <StoryContent>: {{story_content}}
6. <SceneCharacters>: {{characters}}
7. <TaskScene>: {{task_scene}}
8. <TaskScenePlotElement>: {{plot_element}}
9. <TaskScenePlace>: {{place}}
10. <RolePlayHistory>: {{role_play_history}}

B.3 Dramatron

We modified Dramatron's original prompt template in order to adapt their work for writing stories. We removed their example for two reasons: Their work was designed for writing screen play and all systems used for experiment are zero-shot. Below is the prompt we used.

Scene Content Generation

```
Use the following description, write the content of the scene
Place: {{place_name}} + \n + {{place_description}}
Characters: {{characters}}
Plot Element: {{plot_element}}
Summary: {{summary}}
Outline: {{outline}}
```

B.4 Agents'Room

We followed strictly the implementation of the original paper for all implementations.

B.5 LLM Evaluation

Rewrite to Story

You will conduct a side-by-side evaluation. You will be given two system-generated stories. Your task is to compare the two stories and determine which one is better based on the following dimensions:

- Plot: The story should have a recognizable structure, e.g., with a connected beginning, middle, and end. The story should exhibit events and turns that move the plot forward. The story should not have logical or conceptual inconsistencies. Surprising or disruptive elements should be intentional, e.g., they serve the story and do not feel jarring, odd, or out of place.
- Creativity: There should be engaging characters, themes, and imagery. The ideas should not feel generic or bland. There should be avoidance of overly cliched characters and storylines, unintentional tropes, and stereotypes. When used, tropes and cliches should serve a purpose (e.g., comedic effect, twist on a common trope etc). The story should include original elements that were not explicitly mentioned in the prompt.
- Development: Characters and settings should be introduced and contextualized with relevant details that allow the reader to understand their place in the story. Appropriate levels of detail and complexity should be provided to lend the story a feeling of realness and believability.
- Language Use: The language used should feel varied and rich: Variance of sentence structure, verbiage, and vocabulary. The story should exhibit rhetorical, linguistic and literary devices (e.g., ambiguity, alliteration, etc) to create interesting effects. The story should avoid bland or repetitive phrases (unless used intentionally to create a narrative, thematic, or linguistic effect).

Provide a detailed assessment of the two stories in terms of these four dimensions. Conclude your assessment with scores for each dimension using the template below. Do not add any emphasis, such as bold and italics, on your assessment.

[Assessment Ending Template]

Based on my assessment, the better story for each dimension is:

Plot: [A or B or Same]

Creativity: [A or B or Same]

Development: [A or B or Same]

Language Use: [A or B or Same]

Overall: [A or B or Same]

[Story A]

{{story_A_content}}

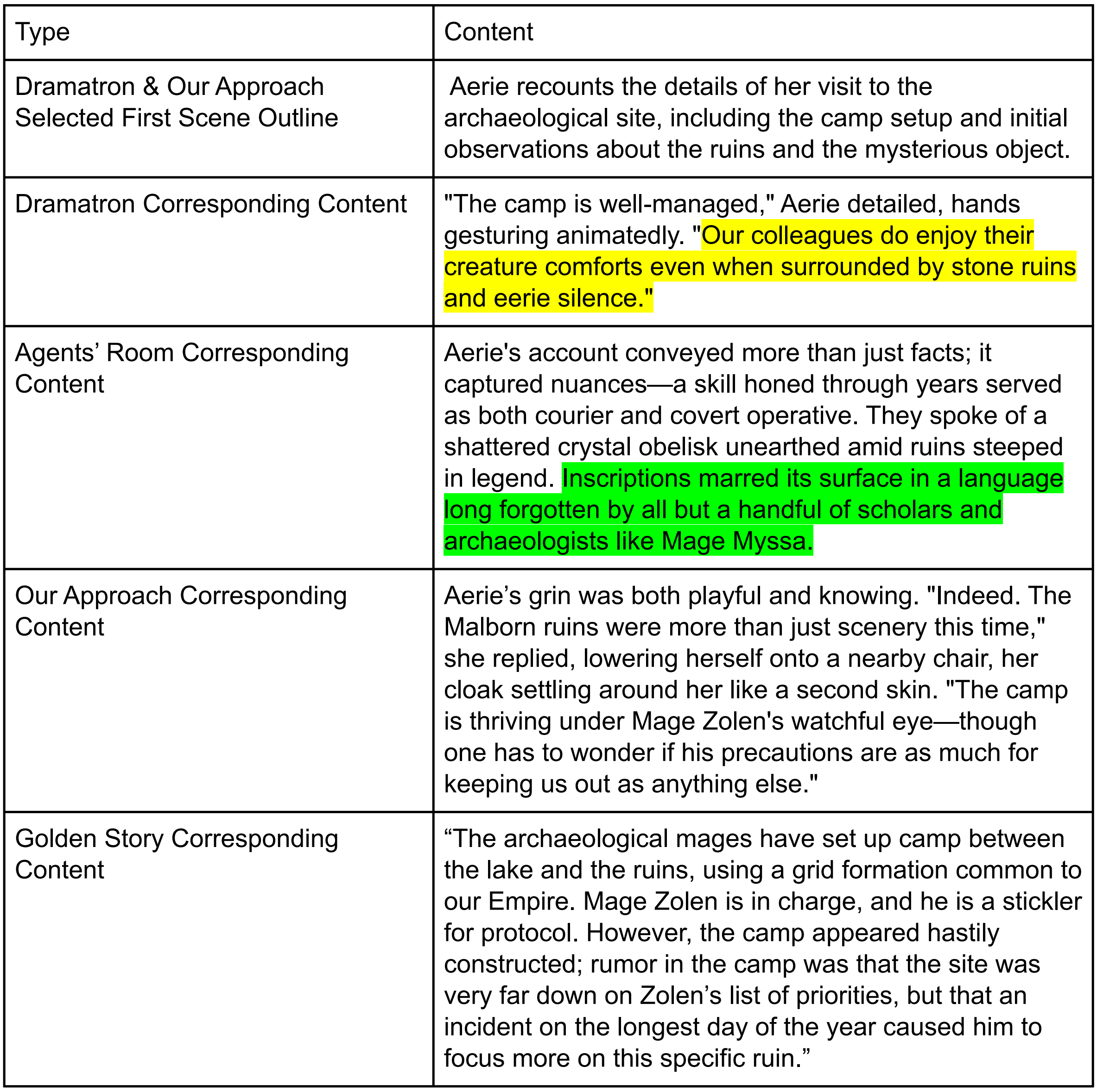
[Story B]

{{story_B_content}}

[Assessment]

C Qualitative Analysis Sample

C.1 Consistency Comparison



Type	Content
Dramatron & Our Approach Selected First Scene Outline	Aerie recounts the details of her visit to the archaeological site, including the camp setup and initial observations about the ruins and the mysterious object.
Dramatron Corresponding Content	"The camp is well-managed," Aerie detailed, hands gesturing animatedly. "Our colleagues do enjoy their creature comforts even when surrounded by stone ruins and eerie silence."
Agents' Room Corresponding Content	Aerie's account conveyed more than just facts; it captured nuances—a skill honed through years served as both courier and covert operative. They spoke of a shattered crystal obelisk unearthed amid ruins steeped in legend. Inscriptions marred its surface in a language long forgotten by all but a handful of scholars and archaeologists like Mage Myssa.
Our Approach Corresponding Content	Aerie's grin was both playful and knowing. "Indeed. The Malborn ruins were more than just scenery this time," she replied, lowering herself onto a nearby chair, her cloak settling around her like a second skin. "The camp is thriving under Mage Zolen's watchful eye—though one has to wonder if his precautions are as much for keeping us out as anything else."
Golden Story Corresponding Content	"The archaeological mages have set up camp between the lake and the ruins, using a grid formation common to our Empire. Mage Zolen is in charge, and he is a stickler for protocol. However, the camp appeared hastily constructed; rumor in the camp was that the site was very far down on Zolen's list of priorities, but that an incident on the longest day of the year caused him to focus more on this specific ruin."

The table above compares content generated by each system corresponding to the same selected outline segment. The yellow highlighting indicates a hallucination produced by the Dramatron approach, resulting in nonsensical content. The green highlighting marks premature references to content and characters intended to appear later in the story.