



**POLITECNICO**  
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE  
E DELL'INFORMAZIONE

# A Scaffolding Architecture for Interactive and Immersive Storytelling with Small Language Models

TESI DI LAUREA MAGISTRALE IN  
COMPUTER SCIENCE ENGINEERING - INGEGNERIA INFORMATICA

Nicolò Giallongo, 10764261

**Advisor:**  
Prof. Franca Grazotto

**Co-advisors:**  
Giulia Valcamonica  
Francesco Piferi  
Francesco Vona

**Academic year:**  
2025-2026

**Abstract:** This thesis presents the design, implementation, and evaluation of a scaffolding architecture for interactive and immersive storytelling using small Large Language Models. The proposed architecture supports the generation process by providing modular components for information extraction, retrieval-augmented generation, and context management, aiming to reduce the computational burden and improve long-term coherence in resource-constrained environments.

The architecture was implemented as a modular system built around the Llama-3.1-8B-Instruct model, and tested against a purely LLM-based baseline in an interactive fantasy storytelling scenario. A small-scale user study was conducted, involving seven tests with five users. Quantitative and qualitative analyses were performed, combining user questionnaires (UEQ, SUS, and custom story evaluation) with automated textual metrics covering lexical diversity, adherence to story outline, adherence to user input, and semantic coherence.

Results indicate that the scaffolding architecture does not significantly degrade the perceived or objective quality of the generated stories, while offering substantial advantages in terms of computational efficiency. Specifically, the architecture maintains both processing time and context token length roughly constant as the story progresses, whereas the baseline system exhibits linear growth in both dimensions. Moreover, evidence suggests that the purely LLM-based approach may begin to degrade after approximately 20 generated scenes. These findings support the conclusion that the proposed architecture is a viable solution for interactive storytelling with small LLMs in low-resource settings, offering a favorable trade-off between story quality and computational cost.

**Key-words:** scaffolding architecture, interactive storytelling, immersive narrative, large language models, small LLMs, resource-constrained environments, context management, narrative generation.

# Contents

|          |                                                                      |           |
|----------|----------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                  | <b>4</b>  |
| <b>2</b> | <b>Related Works</b>                                                 | <b>5</b>  |
| 2.1      | Traditional storytelling structures . . . . .                        | 5         |
| 2.2      | Role-playing games . . . . .                                         | 6         |
| 2.2.1    | Dungeons & Dragons as an Interactive Storytelling Paradigm . . . . . | 6         |
| 2.3      | Interactive storytelling in video games . . . . .                    | 7         |
| 2.3.1    | Visual Novels . . . . .                                              | 7         |
| 2.4      | Storytelling with LLMs . . . . .                                     | 8         |
| 2.4.1    | Commercial Products: AI Dungeon and AI Roguelite . . . . .           | 9         |
| 2.5      | Key Takeaways and Future Perspectives . . . . .                      | 9         |
| <b>3</b> | <b>Implementation</b>                                                | <b>10</b> |
| 3.1      | Architecture components . . . . .                                    | 10        |
| 3.1.1    | Retrieval System . . . . .                                           | 10        |
| 3.1.2    | Information extractor . . . . .                                      | 15        |
| 3.1.3    | Narration Generator . . . . .                                        | 17        |
| 3.1.4    | Hyperparameters choice . . . . .                                     | 17        |
| 3.2      | Application back-end . . . . .                                       | 18        |
| 3.2.1    | Functionalities . . . . .                                            | 18        |
| 3.2.2    | Application design . . . . .                                         | 21        |
| 3.3      | Application front-end . . . . .                                      | 30        |
| 3.3.1    | UI/UX design . . . . .                                               | 30        |
| 3.4      | Additional Notes on the application and the architecture . . . . .   | 33        |
| 3.4.1    | Relation between application and architecture . . . . .              | 33        |
| 3.4.2    | The author perspective . . . . .                                     | 33        |
| <b>4</b> | <b>Testing of the solution</b>                                       | <b>34</b> |
| 4.1      | Limitations of the testing . . . . .                                 | 34        |
| 4.2      | Architecture bypass . . . . .                                        | 35        |
| 4.3      | Evaluation questionnaire . . . . .                                   | 36        |
| 4.4      | Conducted tests overview . . . . .                                   | 36        |
| 4.4.1    | Test A . . . . .                                                     | 37        |
| 4.4.2    | Test B . . . . .                                                     | 37        |
| 4.4.3    | Tests C1 and C2 . . . . .                                            | 38        |
| 4.4.4    | Tests D1 and D2 . . . . .                                            | 39        |
| 4.4.5    | Test E . . . . .                                                     | 39        |
| 4.4.6    | Wrap up and general comments on the tests . . . . .                  | 40        |
| <b>5</b> | <b>Quantitative Results of the Testing</b>                           | <b>40</b> |
| 5.1      | Quantitative analysis of the user testing results . . . . .          | 40        |
| 5.2      | Quantitative analysis of the produced texts . . . . .                | 45        |
| 5.2.1    | Diversity . . . . .                                                  | 45        |
| 5.2.2    | Adherence to story outline . . . . .                                 | 51        |
| 5.2.3    | Adherence to user input . . . . .                                    | 53        |
| 5.2.4    | Coherence . . . . .                                                  | 55        |
| 5.3      | Evaluation of the efficiency of the architecture . . . . .           | 57        |
| 5.3.1    | Waiting and generation time . . . . .                                | 57        |
| 5.3.2    | Context tokens . . . . .                                             | 59        |
| <b>6</b> | <b>Conclusions</b>                                                   | <b>59</b> |
| <b>7</b> | <b>Limitations and Future Works</b>                                  | <b>60</b> |
| 7.1      | The author perspective . . . . .                                     | 60        |
| 7.2      | Limitations on the testing . . . . .                                 | 60        |
| 7.3      | Limitations on the data analysis . . . . .                           | 60        |
| 7.4      | Parameter and hyperparameter tuning . . . . .                        | 61        |
| 7.5      | Model fine-tuning . . . . .                                          | 61        |

|          |                                           |           |
|----------|-------------------------------------------|-----------|
| 7.6      | Used models . . . . .                     | 61        |
| <b>8</b> | <b>Bibliography and citations</b>         | <b>61</b> |
| <b>A</b> | <b>User Questionnaire</b>                 | <b>72</b> |
| A.1      | User information . . . . .                | 72        |
| A.2      | UEQ . . . . .                             | 73        |
| A.3      | SUS . . . . .                             | 73        |
| A.4      | SE . . . . .                              | 74        |
| <b>B</b> | <b>Prompts</b>                            | <b>75</b> |
| B.1      | Scene Generation . . . . .                | 75        |
| B.2      | Action extraction . . . . .               | 78        |
| B.3      | Path taking . . . . .                     | 79        |
| B.4      | Companion joining detection . . . . .     | 80        |
| B.5      | Scene summarization . . . . .             | 81        |
| B.6      | Generation without architecture . . . . . | 81        |
| <b>C</b> | <b>Story Outline</b>                      | <b>83</b> |
| C.1      | Outline correspondence . . . . .          | 83        |
| C.2      | Additional elements . . . . .             | 87        |
| C.3      | General considerations . . . . .          | 91        |

# 1. Introduction

Generative AI has become, in the last 5 years, a widespread phenomenon that permeates the daily lives of many people. The most notorious examples of this are Large Language Models (LLMs), popularized through chatbot applications such as ChatGPT [56], Gemini [72], DeepSeek [22] and many others. Most LLMs are models based on transformers [78, 86], a specific neural network architecture. In this work, we will not engage with the technicalities of how LLMs work. It will suffice for our needs to know the basic mechanism behind these kinds of models. LLMs are at their core probabilistic models: they take a sequence of tokens (roughly corresponding to words or parts of words) as input, called the **context**, and output a sequence of words, each word of which is chosen based on probability sampling over a set of words. The probability of each possible word is computed based on the context and the previously generated words [20, 86, 88]. Although this mechanism may seem simple, the neural network structure of these models allows them to predict very accurately the "right word" to produce. If adequately trained, the neural networks can develop the ability to produce responses to the input that closely mimic human language and give the impression of intelligence. The capabilities of a model depend on various factors, but are strongly correlated with the number of parameters of the neural networks that constitute the model. Modern LLMs usually have no fewer than one billion (1B) parameters, and the larger models available are estimated to have hundreds of billions of parameters [31, 45, 64, 86]. Large models generally perform better than smaller ones [41, 86].

However, smaller models can outperform larger ones when fine-tuned for specific tasks [7, 15, 30]. Fine-tuning consists of training a smaller model on domain-specific data to enhance performance on a targeted activity [66]. Another very important aspect that influences the performance of an LLM is hyperparameter tuning. However, hyperparameter tuning and fine-tuning are usually costly operations (especially the latter [68]) both in terms of time and computational resources. Moreover, these methods apply only to a single model; if one wants to experiment with other models, these procedures must be repeated. In complex scenarios, which require performing a wide range of different tasks, large LLMs still outperform smaller ones.

Another important aspect when dealing with LLMs is prompting: using the right set of input words and the right context can make a significant difference in the performance of an LLM. As noted in recent literature, prompt engineering has emerged as an integral technique for extending the strengths and abilities of Large Language Models to achieve significant performance gains in various Natural Language Processing tasks [21, 28].

The techniques described so far have been extensively explored in the literature and their validity is widely accepted by the scientific community. However, LLMs have structural limitations that these techniques can at best mitigate, not overcome. One of the main limitations of LLMs is their difficulty to maintain a large context. In tasks that require a continuous flow of information, LLMs, especially smaller ones, fail to maintain a coherent context and to extract from it the relevant information needed at each moment [39].

Interactive narrative tasks are a domain in which LLMs show the aforementioned weak points. Interactive story generation requires maintaining a long context and performing different tasks at the same time (extracting information from the user's input, reasoning about the progression of the story, generating a coherent and creative scene). Larger LLMs perform fairly well on interactive storytelling tasks, at least within the span of a few scenes. However, smaller LLMs are limited by their simpler structure. Classic techniques (fine-tuning, hyperparameter tuning, prompt engineering) can significantly improve the performance of small LLMs in narrative tasks, but cannot address the context limitation, and the quality of the produced story reduces scene by scene.

To address the problem of context and improve the quality of the stories produced in interactive narrative tasks, our approach is to employ an external scaffolding application which supports the LLM in the generation of the story.

Our goals in this work are:

- (1) to design a scaffolding architecture external to LLMs that can be used with any model;
- (2) to tailor this architecture for the goal of interactive storytelling;
- (3) to design an application that employs the architecture to offer an immersive interactive narrative experience;
- (4) to test the architecture on a small LLM (less than 10 billion parameters) and see if it actually shows improvement in the immersive interactive storytelling task. The goal is to see whether it is possible to run such an application using limited hardware resources.

Inevitably, the design and implementation of this architecture will include (but will not be limited to) prompt engineering. We will however neither perform fine-tuning nor hyperparameter tuning.

The goal of the architecture is to support the LLM in generating each scene of an interactive story. It does so by providing all the necessary information while avoiding context explosion, and by performing additional operations on both user input and generated output to assist story generation.

## 2. Related Works

Storytelling has always been an important aspect of human life. Theoretical and empirical research indicates that storytelling and narrative activities support the construction of the self [14, 29], the development of autobiographical memory [29], and the acquisition of linguistic and socio-cognitive skills [53]; moreover, storytelling interventions have been linked to positive emotional outcomes such as resilience [62, 63].

Historically, storytelling has developed in tandem with social and technological progress. Originating in orally transmitted epic poetry and later transitioning to written and illustrated texts, it eventually expanded into visual media such as film, demonstrating how narrative forms adapt to the media through which they are conveyed. Over the past fifty years, the emergence of the video-game medium has enabled the popularization of highly interactive storytelling, in which player choices can significantly shape the course of the narrative.

The concept of interactive storytelling, in which the user’s decisions impact the flow of the narration, can be traced back to 1930, with the book-game *Consider the Consequences!*, which allowed the reader to take choices on the story course by jumping to specific pages. The book features three main characters, each with its own ramified storyline, and the impressive number of 43 endings. This idea was later re-used for different books, including the famous series of books *Choose Your Own Adventure*, which sold more than 250 million copies between 1979 and 1998[18].

Another major break-through in the interactive storytelling was the publishing in 1974 of the Table-top role-playing game (TTRPG) *Dungeons & Dragons*, which popularized the genre and is still today the most known example of TTRPG. The basic idea behind *Dungeons & Dragons* (and TTRPGs in general) is for the players to cooperate in the creation of a story. Each player acts as a character in the story, with its own objectives, personality and abilities. Among the players, one takes the role of the Game Master (GM), which guides the other players through the unrevealing of the story.

Video-games represent the ideal medium through which approach interactive storytelling, since video-games are inherently an interactive media. Most of modern video-games allows the player to choose among at least two endings, while some titles specifically features different branches of the main story, which takes into account all the choices taken by the player throughout the game.

Generative AI represents the latest major technological breakthrough, with profound implications for society. Among the domains affected by this emerging technology is storytelling, where Large Language Models (LLMs) have demonstrated notable advances in generating creative narratives, following the users instruction, yet still exhibit potential limitations and biases.

In the following subsections, we break down in detail some of the topics previously introduced, when particularly interesting for the purpose of this work. Moreover, we will analyze the current state of the art, and related works on LLM and storytelling.

### 2.1. Traditional storytelling structures

The study of recurrent structures in narrative works dates back to Aristotle’s *Poetics* and has evolved throughout history through the contributions of various authors. In this section, we retrace the main narrative structures and patterns outlined by different scholars. The source for this overview is the article *Survey on Narrative Structure: from Linguistic Theories to Automatic Extraction Approaches* [11], particularly Section 2. All primary sources referenced in this subsection (mainly books) are cited within that article. The decision to reference only this survey stems from the relatively marginal relevance of this section to the core topic of this work and from the time-consuming nature of consulting all primary sources directly.

Aristotle classifies a narrative into three main parts: the **beginning**, where characters and settings are introduced; the **middle**, where the conflict emerges and the protagonists become aware of the central problem; and the **end**, where the problem is resolved and the protagonists’ situation returns to normal.

Propp identified recurring plot elements in his morphological study of folktales, which he termed *functions*. He defines a function as “an act of a character, defined from the point of view of its significance for the course of the action.” Each function involves characters filling specific roles, the *dramatis personae*. Propp identified seven *dramatis personae*: Hero, Villain, Princess, Dispatcher, Donor, Helper, and False Hero. He also outlined 31 story elements organized into four spheres: the **introduction**, the **body of the story**, the **donor sequence**, and the **hero’s return**.

Todorov and Weinstein further developed these ideas, proposing a five-step structure: **equilibrium**, **disruption**, **realization**, **restored order**, and **equilibrium (again)**.

A similar five-step narrative structure—**Exposition**, **Rising Action**, **Climax**, **Falling Action**, and **Denouement**—was also proposed by Freytag.

Berhe et al. [11] draws a correspondence between these different story structures.

Aristotle’s beginning corresponds to the introduction and equilibrium stages in Propp’s and Todorov’s theories, respectively. Aristotle’s middle aligns with Propp’s body of the story and donor sequence, as well as with

Todorov’s disruption and realization phases. Finally, Aristotle’s end corresponds to Propp’s hero’s return and Todorov’s restored equilibrium.

Chatman, and later Bordwell, divided narrative into *histoire*, the raw content of events, characters, and entities, and *discours*, the manner in which the story is presented. Berger proposed a similar distinction, referring to these concepts as “fabula” and “syuzhet.”

Modern writers—especially in visual media such as films and television series—often draw on Campbell’s theory of the mythological “hero’s journey.” The most common structural techniques used by filmmakers (primarily in Hollywood) are the three-act (III-act) structure, based on Aristotelian principles, and the five-act (V-act) structure, reminiscent of the frameworks proposed by Todorov, Weinstein, and Freytag. An act, as defined by McKee, is a “series of sequences that peaks in a climactic scene which causes a major reversal of values, more powerful in its impact than any previous sequence or scene.”

Recently, the three-act structure has also been adopted by numerous narrative video games, including the acclaimed titles *Baldur’s Gate III*, *Hollow Knight: Silksong*, and *Claire Obscure: Expedition 33*.

Given its simplicity and its widespread adoption across modern narrative works, this research adopts the three-act structure (Setup, Confrontation, and Resolution) as the target structure for the generated story.

## 2.2. Role-playing games

Role-playing games (RPGs) can be defined as structured play activities in which participants assume and enact fictional characters within a **shared narrative setting** [85], making decisions governed by a system of rules [49] that **collaboratively drive the progression of the story** and express player agency [75].

Role-playing games (RPGs) manifest across a variety of media. Many single and multiplayer video games incorporate role-playing elements, and historical reenactments similarly rely on participants assuming specific roles within reconstructed events. However, the most prominent category remains the Tabletop Role-Playing Games (TTRPGs), exemplified by *Dungeons & Dragons*, which has evolved over more than fifty years through five editions.

Empirical research indicates that RPGs confer a range of benefits, particularly for populations with special needs, such as autistic individuals [8], as well as for children. In educational contexts, TTRPGs have been shown to support the development of critical social-emotional learning (SEL) skills by engaging students in morally relevant scenarios that are uncommon in everyday life [71].

### 2.2.1 Dungeons & Dragons as an Interactive Storytelling Paradigm

*Dungeons & Dragons* (hereafter D&D) is widely regarded as the most prominent example of a tabletop role-playing game (TTRPG). First published in 1974, D&D has exerted a substantial influence on Western popular culture; by the 1980s it was already considered a cornerstone of American “nerd” culture, helping to shape a shared imaginary centered on fantasy, adventure, and collaborative storytelling [27, 37, 55].

Over the past decade, D&D has experienced a significant resurgence in mainstream media: frequent references in mass-market productions (notably the Netflix series *Stranger Things*), the expansion of its transmedia ecosystem, and the growing popularity of live-streamed or online sessions have all contributed to its transition from a niche hobby to a widely recognized cultural phenomenon. A further indication of this renewed popularity is the release, in 2023, of the feature film *Dungeons & Dragons: Honour Among Thieves*, which was positively received by audiences and critics alike [2]. In the same year, the video game *Baldur’s Gate 3*, which makes extensive use of the D&D rule-set and narrative setting, won several Game of the Year awards [1, 3, 42].

Given this cultural impact, the aforementioned benefits of TTRPGs, and the wide amount of existing literature, we chose D&D as the main inspiration for the type of interaction that our interactive storytelling application aims to reproduce.

In this subsection, we describe how the interactive storytelling mechanism works in D&D, and how we aim to translate it into our demo application. We limit the discussion to the narrative and storytelling aspects of the game, intentionally leaving aside the complex rule-set that governs combat encounters. Although these mechanics constitute a substantial part of D&D as a whole, they are irrelevant in the context of this study. The sources for the following description are the latest version of the official D&D Player’s Handbook [83] and the Dungeon Master’s Guide [82], complemented with extensive direct experience with the system.

The role-playing game system of D&D revolves around two primary roles: **Player** and **Game Master** (GM, or Dungeon Master).

- **Player.** Each Player acts as an “actor” within the game’s narrative setting. Players create a **Character** that becomes part of the group of adventurers (the **party**) who serve as the protagonists of the story. Each Player defines their Character’s appearance, personality, backstory, moral compass, and objectives. Through their Character’s actions, choices, and interactions, **Players directly influence the unfolding**

**narrative.** Unlike choose-your-own-adventure books or traditional RPG video games, Player agency in D&D is theoretically unlimited rather than restricted to predefined options.

- **Game Master.** The GM acts as narrator, director, and—at times—author of the story experienced by the Players. The GM’s responsibilities include describing environments, events, and characters; reacting to the Players’ actions; and narrating the consequences of those actions. The GM provides vivid descriptions that help Players visualize scenes, explains relevant aspects of the fictional world (such as lore or cultural norms), and role-plays Non-Player Characters (NPCs) who interact directly with the Players’ Characters (PCs).

According to the Player’s Handbook, gameplay unfolds through the following recurring pattern:

1. **The GM describes the scene.** The GM presents the location, sensory details, ongoing events, and relevant context for the situation in which the PCs find themselves.
2. **Players describe what their Characters do.** Each Player declares the actions their Character attempts to perform, which may include interacting with NPCs, examining the environment, or engaging in combat. These actions often require the use of Character abilities or dice rolls.
3. **The GM narrates the outcome.** Based on the declared actions, game logic, and (when applicable) the results of dice rolls, the GM describes the consequences of success or failure, updating the narrative accordingly.

This loop repeats throughout a game session, enriched by dialogue between PCs and NPCs, combat encounters, puzzles, and other forms of interaction. Despite these variations, the fundamental three-step structure provides a simple yet effective template for interactive storytelling.

In the context of the application developed for this research, the LLM (and the surrounding system architecture) takes on a role analogous to that of the GM (in its Narrator role), while the user assumes the role of the Player. Accordingly, the interactions between the user and the system follow the three steps outlined above. However, the application does not attempt to reproduce the full D&D ruleset; rather, it draws inspiration from the GM–Player roles and their underlying interaction loop. It is important to note that this structure is not unique to D&D: it represents the foundational interaction pattern of most TTRPGs. D&D was selected as a reference due to its broad recognition, its frequent appearance in academic literature, the familiarity of the researchers with the system, and the fact that the Player’s Handbook explicitly describes this interaction pattern.

## 2.3. Interactive storytelling in video games

Since the commercial success of Atari’s *Pong* in 1972, the video game industry has undergone a remarkable evolution, experiencing exponential economic growth. Today, it constitutes one of the largest and most profitable sectors within the global entertainment and creative media landscape. Even when excluding secondary revenue streams, global video game revenues surpass the combined revenues of the cinema and music industries [61].

The expansion of the video game industry is not limited to its economic footprint. It has also been a major driver of technological innovation. Video games have historically pushed the development of high-performance computing hardware, much of which later finds crucial applications in other fields [4]. A notable example is the GPU: originally conceived for graphics processing and popularized primarily through gaming, GPUs have become a fundamental component in modern artificial intelligence workflows [47]. Similarly, areas such as real-time graphics rendering [5, 57] and virtual or augmented reality technologies have significantly benefited from the technological demand generated by the video game industry.

Beyond technological innovation, one of the most relevant developments within the medium is the rise of interactive storytelling. Interactivity in video games typically manifests through player engagement with game mechanics to overcome challenges, through interactions with the surrounding virtual environment, or through narrative interaction—whereby the player’s actions influence the unfolding of the story. These paradigms often converge. A paradigmatic example is the acclaimed 2015 RPG *Undertale* by Toby Fox, in which the way players approach combat encounters fundamentally shapes the narrative experience.

For the purposes of this work, we focus exclusively on narrative interaction. Among the different video game genres that employ this paradigm, Visual Novels (VNs) exemplify it in its purest and most explicit form.

### 2.3.1 Visual Novels

Visual Novels are a narrative-focused genre that emerged primarily in Japan and later achieved international popularity through titles such as *Steins;Gate*—which also inspired an anime adaptation—and *Doki Doki Literature Club!*. Although they share certain features with role-playing games, Visual Novels are generally considered a distinct genre, characterized by minimal gameplay, a strong emphasis on textual and dialogic interaction, and branching storylines that lead to multiple possible endings.

The core of the VN experience lies in character interactions, emotional engagement, and player-driven narrative progression. Due to their predominantly textual nature, however, Visual Novels face inherent limitations: player

interaction is constrained to a finite set of predefined dialogue options, which restricts the expressive range of conversations with non-player characters (NPCs). Moreover, supporting a large number of branching storylines requires substantial authorial effort, as every branch must be manually written and integrated.

Given these characteristics, Visual Novels are particularly well suited to benefit from the integration of large language models (LLMs). Their reliance on text-based interaction, player choice, and narrative branching aligns closely with the generative capabilities of modern LLMs, which can dynamically produce dialogue, adapt narrative flow, and expand the spectrum of possible user interactions. For these reasons, VNs serve both as an inspiration and as an ideal application domain for the interactive storytelling system developed in this work.

Commercial products such as *AI Dungeon* and *AI Roguelite* already apply the use of LLMs to a contest similar to a VN (including also RPG elements). These two examples will be discussed later.

## 2.4. Storytelling with LLMs

Large Language Models (LLMs) provide powerful tools in text generation. Storytelling naturally falls in this domain, and hence a lot of research efforts has been drawn to the effective use of LLMs for quality storytelling. However, sustaining long term memory and producing coherent text within an overall theme has proven a challenging tasks for LLMs [39]. Also, Creative Natural Language Generation lacks reference texts and heavily relies on user evaluations [10], making more difficult to both train and evaluate Storyteller LLMs.

Bensaid et al. [10] attempted to address this issue by introducing a re-ranking pipeline. Their system relies on a GPT-2-based model fine-tuned to generate multiple candidate narratives, which are subsequently ranked according to several metrics, including readability, positive sentiment, diversity, simplicity, coherence, and “tale-likeness.” The top three outputs are then presented to the user. Conceived as a creative assistant for human storytelling, the model supports idea generation and enables co-authoring with the user. Additionally, the system integrates an image retrieval module that selects visual content related to the generated text. Although it was trained to produce children’s tales, its intended end-users were adult authors.

A similar strategy was adopted by Nichols et al. [52], who also employed a fine-tuned GPT-2 model in combination with a ranking mechanism to select the final narrative output. In this system, a GPT-2-based ranker with a classification head assigns a scalar score to the last token of each generated candidate, leveraging the unidirectional nature of GPT-2 to encode the entire sequence. A softmax over these scores determines the preferred outputs. The model was designed for collaborative storytelling through turn-taking with a human user. Although it could respond creatively to user inputs, it remained susceptible to narrative incoherence and lacked a well-defined story structure.

The lack of story structure in LLM-generated stories is a well known problem. Research focused on solving it by means of scaffolding architectures that supports text generation.

Simon et al. [67] used a planner based on Planning Domain Definition Language (PDDL) to create a story structure. Each plan action is fed to the LLM, along with the previous actions and corresponding model output, other than a starting "hidden prompt" containing general information on the story or common sense information. While this approach proved successful on providing a story structure to the LLM, it requires the manual translation of the story to PDDL, which makes this architecture not suitable for imaginative and/or interactive storytelling.

Similarly, Yoo and Yun-Gyung [84] employs a planning-based approach in which a graph planner is used to guide the output of the LLM. The framework is grounded in the classical narrative distinction between fabula (the complete chronological sequence of story events), syuzhet (the narratively arranged subset of events selected to achieve specific storytelling goals), and discourse (the final linguistic realization of the narrative). In this model, the syuzhet functions as the story plan. The system first constructs the fabula, then selects events based on a relevance score to generate a preliminary story skeleton. This sequence is subsequently reordered to form the syuzhet, which is then provided to the LLM to produce the final narrative (discourse). The study focused on the skeleton selection, and thus did not produce results on the actual discourse generation.

Wang et al. [81] introduce a distinct approach that integrates character simulations, author-defined high-level plot structures (referred to as “abstract acts”), and player actions to produce cohesive and emergent narratives in story-driven video games. Within this framework, the story dynamically adapts to changes in the game environment, guided by the abstract acts and informed by LLM-based character simulations, thereby enabling the generation of coherent and contextually responsive narratives.

Góngora et al. [35, 36] developed a scaffolding-based architecture designed to track the state of the world and update it according to the user’s actions. Their system, named PAYADOR, aims to emulate the activity of a Game Master in a TTRPG. In PAYADOR, a class `World` maintains the world state, organizing it into four main types of information (Items, Locations, Characters and Puzzles). To update the world state, the system maps the user’s textual inputs to a predefined set of *transformations*, which are atomic pre-programmed effects capable of modifying the world. The key innovation introduced by PAYADOR is that *transformations* do not encode the player’s possible actions, but rather the *effects* that those actions produce. For example, consider the

*transformation* **Unblocked locations**, which removes an obstacle that previously prevented access to another area. Imagine two locations separated by a closed door: the user may choose to use a key, pick the lock, pry it open with a crowbar, or break it down. Although these actions differ, they all lead to the same outcome from the perspective of the world state: the elimination of the obstacle. PAYADOR models all such cases through the **Unblocked locations** transformation.

This design allows PAYADOR to remain flexible with respect to player decisions while still providing a structured and consistent mechanism for representing the world state and its updates.

#### 2.4.1 Commercial Products: AI Dungeon and AI Roguelite

Beyond academic research, commercial products also explore LLM-based storytelling.

AI Dungeon is an online narrative RPG that uses LLMs to dynamically generate story content. A limited version is available for free, while different subscription tiers unlock more advanced features. AI Dungeon is exclusively text-based. At the start, the user selects a genre (fantasy, mystery, etc.), a character archetype, and provides a character name. The narration then begins, introducing the player to the story. It is delivered in a colloquial register and directly addresses the player, mimicking interactions between a Game Master and a Player in a TTRPG. The story lacks a predefined goal, and the user simultaneously assumes the role of Player (controlling character actions) and author (shaping narrative events).

AI Roguelite is likewise a narrative RPG featuring AI-generated content, including both text and images. Unlike AI Dungeon, it incorporates a wide range of RPG mechanics such as combat, items, and NPCs, supported by a complex user interface; it is therefore not a purely textual experience. Here too, the narration directly addresses the player, and the only clear objective is survival.

Van der Torre [77] conducted an analysis of user reviews on Steam and Reddit for both AI Dungeon and AI Roguelite. His findings indicate that, although users respond positively to the generative capabilities of these applications, they also report that LLMs frequently produce incoherent or repetitive narratives. While this analysis has certain limitations, it aligns with known constraints of LLMs and with direct experience using the systems, and can thus be cautiously considered. Furthermore, Góngora [35] performed qualitative tests on AI Dungeon, showing that the system is unable to correctly track the state of the world.

### 2.5. Key Takeaways and Future Perspectives

In this section, we summarize the key takeaways from the literature discussed above that have influenced our work, and we briefly outline the general idea of our approach, highlighting the contributions of each work or concept introduced throughout Section 2.

We are interested in building a scaffolding architecture to assist an LLM in the generation of narrative within the context of fantasy interactive and immersive storytelling. By "interactive", we mean that the user's choices have a meaningful impact on the generated story. By "immersive", we mean that the user adopts a personal point of view within the story, rather than an omniscient one. This key feature distinguishes the scope of our work from that of co-authoring: the user of our system will not have control over the high-level plot of the story; rather, they will shape the story "from the inside" through interactions with story elements. This scenario shares a large number of similarities with that of TTRPGs. In particular, since fantasy is our target genre, we focused on the most prominent example of this kind of experience, namely *Dungeons & Dragons*. By examining the game's ruleset, we extracted a basic skeleton of its narrative interaction paradigm, which is based on a **description-reaction-description** sequence between the Game Master and the Player. Similarly, in our application, the interaction between the Narrator (the LLM and the architecture) and the User will adopt a congruent turn-taking (description-reaction-description) paradigm. Our focus will be to create a "narrator system" that produces *discourse*, and we will assume that the authoring task (the *histoire*) is produced by an external agent, be it human or artificial. Our application takes inspiration from the work of Góngora et al. [35, 36] for the maintenance of a world state and the use of generalized actions to encompass all possible user approaches. However, as will be better explained in Section 3 (specifically in Sections 3.1.2 and 3.2.1), we want to account for the possible failure of actions depending on the character's characteristics, to enhance immersion. Finally, while our primary mode of interaction remains textual, we will introduce additional interaction mechanics to improve the user experience and increase the number of strategic choices at their disposal. Ultimately, this synthesis guides the design of an architecture that prioritizes immersion and agency, while leveraging the strengths of existing approaches.

### 3. Implementation

Considering the goals expressed in Section 1 and the considerations expressed in Section 2, we now proceed to illustrate the design and implementation of the scaffolding architecture and of the application that employs it. To emphasize the relative independence of the architecture from the application that uses it we divided the implementation into two separate back-ends accessible through APIs, plus a front-end layer for user interaction.

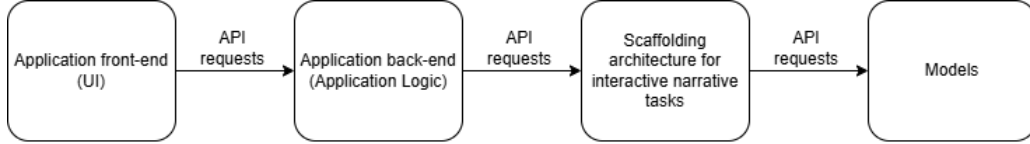


Figure 1: High-level architecture of the solution.

As shown in Figure 1, the high-level architecture of the whole solution is composed of four main components, which communicate through API calls. In the following list we will describe them briefly:

- 1. **Models.** This is the layer that hosts and manages the models used by the solution. We used LM studio for this component of the solution. LM studio is a desktop application that allows to run locally AI models, especially LLMs. Other than offering many useful tools to download, manage and supervise models, the key feature that makes LM studio suitable for our solution is that it allows API calls to the models. This makes it possible to easily load and use models in a completely automatic way.
- 2. **Scaffolding architecture.** This is the architecture at the core of this work. Its role is to invoke various models from the Models component (among which the story generation model) and produce data useful for the functioning of the interactive application. It mediates between the application and the LLMs. The functionalities that this component offers are Retrieval Augmented Generation (RAG), information extraction from the user input, information extraction from the output text, and finally generation of the story scenes.
- 3. **Application back-end.** This component constitutes the real application and its logic. It invokes the functions of the scaffolding architecture through API calls, and uses the data received to perform operations such as updating the application state, invoking other operations from the architecture or outputting the data to the front-end.
- 4. **Application front-end.** This component constitutes the UI. The user will interact with this component and the various interactions will cause invocations to the application back-end.

In the following subsections we explore the list items from 2 to 4 in detail.

#### 3.1. Architecture components

In this subsection we discuss of the components of the architecture which will directly interact with the Models.

##### 3.1.1 Retrieval System

Retrieval-Augmented Generation (RAG) is an architectural framework that integrates the parametric knowledge of a Language Model with external, non-parametric knowledge stored in documents, with the goal of improving model performance and reducing hallucinations, particularly in tasks such as question answering and fact-checking [12, 16, 34]. A RAG system can be conceptually divided into a **Retrieval** component, responsible for retrieving documents (or document segments) relevant to the user query, and a **Generator**, typically an LLM, which produces a response grounded in the knowledge returned by the retriever [34, 65].

The goal of a Retrieval system is to retrieve and rank documents based on the relevance with respect to an input query. The output will be a ranked list containing the top-k documents relevant to the query i.e. the k documents that are more related to the query lexically and/or semantically, ordered from the most to the less relevant.

Within the scope of the architecture and application presented in this work, the Retrieval component is tasked with providing background knowledge about the narrative setting, including character descriptions, locations and narrative context. This information falls within the domain of the *fabula* or *histoire*, providing relevant context to the LLM for scene generation. The goal is to design a fast, low-latency retrieval system capable of operating over a medium-sized document corpus, returning the top-7 most relevant documents for each scene to be generated. Rather than targeting state-of-the-art retrieval performance, the focus of this retrieval system lies in achieving a practical trade-off between retrieval quality, latency, and computational resource consumption.

An extensive review of the literature and existing systems has been conducted to identify the solutions best

aligned to our goal. In the following subsections we outline the two main aspects on which we focused our research efforts.

**Hybrid Retrieval Mechanism** The core component of any retrieval system is the **retrieval mechanism**, which maps a user query (or its representation) to a ranked list of the most relevant documents or document chunks. In practice, the retriever typically returns the top- $n$  most relevant documents, typically with  $n \in [100, 500]$  (depending on the size of the corpus).

The literature broadly identifies three main retrieval paradigms [12, 16]:

- **Lexical (or Sparse) Retrieval:** based on lexical matching between query terms and documents. Documents are preprocessed through tokenization, stopword removal, punctuation removal, case normalization and lemmatization, and then indexed using an inverted index with posting lists, which map each term to the documents in which it appears. At query time, the query undergoes the same preprocessing steps, and candidate documents are retrieved based on term matches. Each document is assigned a relevance score computed by a weighting function, traditionally based on TF-IDF. In practice, the most widely used scoring function is BM25, a probabilistic ranking model that improves upon classical TF-IDF by incorporating term saturation and document length normalization.
- **Semantic (or Dense) Retrieval:** based on semantic similarity between queries and documents. Documents (or their chunks) are projected into high-dimensional vector representations, called embeddings, which capture their semantic content in a continuous vector space. Queries are embedded using the same model, transforming retrieval into a  $k$ -nearest neighbor search problem. The embedding model plays a central role in this paradigm, as its representational quality directly influences retrieval performance. In this work, we take into consideration *embeddinggemma300*, a state-of-the-art embedding model that, at the time of writing, outperforms other models under 500M parameters across most benchmarks, while also remaining competitive with larger models [25, 50, 79].
- **Hybrid Retrieval:** combines sparse and dense retrieval approaches. Sparse retrieval assigns high relevance scores to documents containing exact query terms, particularly rare terms, but fails to capture semantic similarity when relevant documents use different wording. Dense retrieval, while effective at capturing semantic relationships, may lose sensitivity to exact phrasing and lexical cues present in the query. Motivated by the complementary strengths of these approaches, hybrid retrieval architectures combine sparse and dense signals to improve robustness. Empirical evidence shows that hybrid systems consistently outperform purely sparse or purely dense retrieval methods across both in-domain and out-of-domain benchmarks [12, 43, 65, 70].

Based on these considerations, we adopt a hybrid retrieval architecture combining BM25 for sparse retrieval, *embeddinggemma300* for dense representations of documents and queries, and the Annoy library for vector indexing and Approximate Nearest Neighbor (ANN) search. Annoy was selected due to its simplicity of use, light-weightness and the efficiency and scalability of the ANN algorithm. [16, 23, 69, 89, 90]. Both the sparse and dense retrieval systems output a ranked list of the top- $n$  documents (with  $n = 100$ ). These two rankings need to be fused to obtain the output of the hybrid system. Among the various fusion strategies proposed in the literature, two methods are particularly popular due to their simplicity, low computational requirements, and adaptability to different scenarios: Reciprocal Rank Fusion (RRF) and convex score combination [13].

Reciprocal Rank Fusion (RRF) consists of re-ranking documents based solely on their positions in the individual sparse and dense rankings. This method does not require score normalization, nor does it rely on the actual relevance scores produced by the retrieval systems. Instead, it uses only the rank positions of documents in each ranking.

While this makes RRF particularly robust and easy to apply, it may lead to a potential loss of information, as differences in score magnitude between documents are ignored. The RRF score for a document  $d$  is computed as:

$$\text{RRF}(d) = \sum_{i=1}^N \frac{1}{k + \text{rank}_i(d)}$$

The convex combination method consists of combining sparse and dense relevance scores through a weighted sum controlled by a parameter  $\alpha$ :

$$\text{CombinedScore}(d) = \alpha \cdot \text{SparseScore}(d) + (1 - \alpha) \cdot \text{DenseScore}(d)$$

This method requires the normalized scores from the sparse and dense rankings. However, [13] found that the choice of the normalization function has no significant impact on the fusion effectiveness. For simplicity, we adopt the min-max normalization, defined as follows:

$$\text{MinMaxNorm}(d) = \frac{\text{Score}(d) - \min(D)}{\max(D) - \min(D)}$$

where  $D$  is the collection of scores from the top- $n$  document ranking.

The results reported in [13] show that convex score combination with  $\alpha \in [0.6, 0.8]$  consistently outperforms RRF across both in-domain and out-of-domain tasks. These findings motivated our choice of the convex combination approach. Although the parameter  $\alpha$  can be efficiently learned through training [13], given the out-of-domain nature of our task and the secondary role of the retrieval system within the overall architecture, we adopt a fixed value of  $\alpha = 0.7$ . This choice is supported both theoretically—by assigning greater weight to the more robust lexical component—and empirically, as it has been shown to perform well in zero-shot scenarios. This initial choice will later change in a value of  $\alpha = 0.85$ . This was due to the particularly restricted corpus of our toy example, on which semantic matching proved less effective and more prone to error.

**Re-ranking** Both sparse and dense retrieval methods fail to capture deep and fine-grained interactions between queries and documents. As a consequence, the resulting rankings may be imprecise, with highly relevant documents occupying lower positions than expected. More sophisticated ranking approaches—such as bi-encoders, cross-encoders, and LLM-based methods—are capable of modeling richer query–document interactions, but at the cost of significantly higher computational requirements.

This trade-off has led the research community to adopt a two-stage retrieval pipeline. In the first stage, a lightweight retrieval mechanism selects the top- $m$  candidate documents (typically  $m \in [20, 50]$ ) using sparse, dense, or hybrid retrieval methods. In the second stage, a re-ranking mechanism applies a more computationally expensive but more accurate ranking model—such as a bi-encoder, cross-encoder, or LLM-based approach—to reorder the top- $m$  documents and select the final top- $k$  documents (usually  $k \in [5, 10]$ ) to be passed to the generator. Multiple studies have found that incorporating a re-ranking stage in a RAG pipeline consistently improves the performances of the RAG system [12, 54, 60]. Furthermore, empirical evidence indicates that cross-encoders consistently outperform bi-encoders, particularly in in-domain settings [70], while remaining competitive with—and often surpassing—LLM-based re-ranking approaches, especially when time and computational constraints are taken into account [24]. While BERT-based cross-encoders achieve state-of-the-art performance, they require a separate forward pass for each query–document pair, which often introduces computational bottlenecks in large-scale or low-latency systems. Given that the main focus of our RAG system is to achieve low-latency and low resources consumption, we opted for a different model: the jina-reranker-v3 model. This is a 600M parameters model which outperforms all the other models with less than 1B parameters in most MTEB benchmarks, and stays competitive even compared to significantly larger models [25, 50, 80]. Moreover, its architecture applies causal attention between the query and all candidate documents in the same context window, enabling rich interactions before extracting contextual embeddings from each document’s final token, and allowing to re-rank multiple documents in a single pass [80]. While jina-reranker-v3 is slightly less robust in zero-shot settings with respect to embeddinggemma300 used as a bi-encoders [25, 50], the architectural advantages, consistently better performances and its very nature of cross-encoder makes jina-reranker-v3 a more suitable and robust choice for the re-ranking stage of our system.

**Wrap up** To conclude, with the goal of designing a fast, low-latency, lightweight, and well-performing RAG system, we define the following retrieval pipeline:

- Documents are tokenized using a spaCy-based tokenizer and indexed in an inverted index using the BM25L scoring function.
- Documents are embedded using *embeddinggemma300*, and the resulting embeddings are stored in an Annoy index built with 10 trees.
- Upon receiving a query, it is both tokenized (using spaCy) and embedded (using *embeddinggemma300*). Each representation is used to compute, respectively, sparse and dense relevance scores for the documents, producing two ranked lists.
- The relevance scores of the top-100 documents from both sparse and dense rankings are normalized using min–max normalization and fused via convex combination with  $\alpha = 0.85$ .
- Documents are ranked according to the fused scores, and the top-20 candidates are passed to the *jina-reranker-v3* model for re-ranking.
- After re-ranking, the top-7 documents constitute the final output of the retrieval system.

We will, from now on, refer to this retrieval system as Hybrid Retriever with Reranking (HRR).

**Testing of the HRR** With the intent of assessing the capabilities of the HRR, we designed a brief testing campaign. Given the non-central role of the retrieval module in the overall architecture, the objective of this evaluation was not to provide an extensive comparison with state-of-the-art retrieval systems, but rather to verify whether the HRR produced acceptable results, without expending too much time and computational resources on the testing process. To this end, we adopted a step by step evaluation strategy in order to identify potential bottlenecks and assess the contribution of each component. With this goal, we selected the following

metrics for evaluation:

- **Recall@k**. This metrics measures the proportion of relevant documents that are retrieved within the top-k ranked documents. The formula of the recall@k is:

$$\text{Recall@k} = \frac{|\text{Relevant docs in top-k}|}{|\text{Total relevant docs}|}$$

The ideal value of recall@k would be 1 (all relevant document are in top-k). However, this is impossible if  $k < |\text{relevant docs}|$ , and so the recall is typically smaller for smaller k. In our testing procedure, Recall@k was measured at each stage of the retrieval pipeline to monitor how many relevant documents were retained throughout the ranking process.

- **MRR@k**. This metrics is the average, across the queries, of the Reciprocal Rank at k (RR@k) which measures at which position in the rank is found the first relevant document for the query. The formula of the RR@k is:

$$\text{RR@k} = \frac{1}{|\text{Rank of the first relevant document}|}$$

The RR@k has a value of 1 if the document is at first place, of 0.5 if it is second and so on. Therefore the MRR@k is strongly penalized when the first relevant document appears at lower ranks. In our evaluation, we computed MRR@7 on the final top-7 ranking. Together with Recall@7, this metric provides an indication of both coverage and early precision of the retriever.

For the evaluation, we selected the BEIR benchmark. BEIR (Benchmarking IR) provides a homogeneous evaluation framework for diverse sentence- and passage-level information retrieval tasks [73]. It enables cross-domain evaluation of retrieval models under a unified experimental setup. BEIR comprises 18 heterogeneous datasets covering 9 different information retrieval tasks.

For the purposes of our study, we evaluated the HRR using two of these datasets:

- **Google’s Natural Questions (NQ)**. NQ belongs to the Open-Domain Question Answering task. It consists of questions posed by real users, paired with relevant passages extracted from Wikipedia. Each question is associated with one or more relevant passages containing the answer[44].
- **FEVER**. FEVER is a dataset designed for automated fact verification. It consists of claims generated by modifying sentences from Wikipedia pages, along with evidence extracted from Wikipedia to support or confute the claims. The dataset is conceived to label a claim as **supported**, **refuted** or **not enough info**[74].

The task of the HRR is to retrieve (from the context document) information relevant to the user’s input in order to provide the narrator LLM with grounded contextual information upon which to build the narration. Hence, Question Answering and Fact Verification constitute suitable proxy tasks for evaluating the retriever, as both require identifying relevant textual evidence given a natural language query.

In BEIR, these datasets are reformulated to provide a homogeneous benchmark specifically for retrieval evaluation. The original downstream objectives (answer extraction for NQ and claim classification for FEVER) are discarded, and only document relevance with respect to the query is considered. This reformulation allows us to compare results across datasets originally designed for different tasks, while focusing exclusively on the performance of the retrieval component.

The BEIR versions of NQ and FEVER consist of large-scale corpora comprising millions of documents and thousands of queries. To reduce computational cost and enable faster experimentation, we randomly sampled 100 queries from each dataset together with their corresponding relevant documents. To this set, we added 5000 randomly selected non-relevant documents. The retrieval pipeline was then evaluated on this reduced subset. The obtained results are summarized in Table 1

### Retrieval system testing results

|              | BM25:<br>recall@100 | dense:<br>recall@100 | hybrid:<br>recall@20 | reranker:<br>recall@7 | reranker:<br>MRR@7 |
|--------------|---------------------|----------------------|----------------------|-----------------------|--------------------|
| <b>NQ</b>    | 0.9878              | 0.1697               | 0.9036               | 0.8993                | 0.9109             |
| <b>FEVER</b> | 0.9907              | 0.3783               | 0.9606               | 0.9606                | 0.9850             |

**Table 1:** Results of the testing of the HRR with the sample from datasets NQ and Fever. The table shows the step of the pipeline and the corresponding metric measured.

As expected, the BM25 sparse retriever performs strongly in this open-domain setting, whereas the dense retriever exhibits significantly lower recall. This is a well known result in the literature: sparse retrieval often outperforms dense retrieval in zero-shot scenarios without domain-specific fine-tuning. The purpose of the dense retriever in our architecture is not that to boost the overall retriever performance, but rather to complement the sparse retriever by recovering possibly relevant documents not captured through purely lexical matching. In both datasets we notice a slight reduction of the recall of the hybrid rank compared to the BM25 recall. However, this reduction is relatively small (respectively 8% and 3%) and is expected due to the smaller cutoff value  $k$ . Also at the reranking stage, we can observe a slight reduction of the recall on the NQ dataset. However the decrease remains limited (less than 1%) and is consistent with the progressive reduction of  $k$  across the pipeline. The MRR for both datasets is higher than 0.9, which means that the most relevant document is often at first place and rarely at second place or lower in the rankings. This suggests that the reranker is effective in promoting highly relevant documents to the top of the final ranking. From these results, the system might seem highly performative. However, these data are not enough to actually evaluate the performance of the system. The experimental setup is intentionally simplified: the evaluation is conducted on a limited sample of queries and a substantially reduced corpus. This controlled setting lowers the overall difficulty of the task. Extensive evaluations on the full (or different) datasets, as well as additional, more precise metrics might reveal weaknesses under more challenging conditions. However, the goal of this section is not to provide a comprehensive benchmarking study of the system in various settings, but rather to qualitatively verify if the retrieving pipeline operates as intended. Results indicate that the system consistently retrieves relevant documents, and ranks them at high positions, which is sufficient for the purposes of this work.

To further validate these results, we conducted a brief research to find previous works which performed more extensive testing on the BEIR corpus, hoping this could help shed more light on our findings. The work of Kamaloo et al. [40] consists of a deep and complete evaluation across all the BEIR datasets of five retrieval models: two learned dense models (*TAS-B* and *Contriever*), two learned sparse models (*uniCOIL* and *SPLADE*), and *BM25* used as baseline.

The results obtained regarding the Recall@100 from these models on the NQ and FEVER datasets are summarized in the following Table 2:

Retrieval system testing results

|              | BM25  | uniCOIL | SPLADE | TAS-B | Contriever |
|--------------|-------|---------|--------|-------|------------|
| <b>NQ</b>    | 0.760 | 0.833   | 0.930  | 0.904 | 0.925      |
| <b>FEVER</b> | 0.908 | 0.912   | 0.935  | 0.894 | 0.947      |

Table 2: Results from the paper *Resources for Brewing BEIR: Reproducible Reference Models and Statistical Analyses*. We report only the results relative to the datasets NQ and FEVER, and only the Recall@100 metric.

Comparing these results with our own, we can immediately notice how, except for TAS-B, all models have better performances on FEVER than on NQ. Hence, this pattern is not limited to the HRR. Upon further examination of the data, we notice that the Recall@100 score for BM25 shows a significant mismatch between our results and those from [40]. In particular, our results are higher than those found by Kamaloo and their team. This mismatch is perfectly explained by the fact that we used a subset of the corpus. Specifically, we used around 5,000 documents for both datasets, while the whole corpora comprise (respectively for NQ and FEVER) 2,681,468 and 5,416,568 documents. This reduction of the corpora greatly simplifies the retrieval task, boosting the Recall@100. This indicates that the very high scores achieved by our system would not be reached on the original NQ and FEVER corpora in BEIR.

Given these results, we could compute an optimistic estimate of the final Recall@7 our system would exhibit if tested against the full NQ and FEVER corpora. The reduction in recall between BM25 and the final output is 8.96% for NQ and 3.04% for FEVER. If we take the (optimistic) assumption that this percentage reduction would remain the same if the system were tested against the whole corpora, we obtain an estimate of  $\text{Recall@7} = 0.692$  for NQ and  $\text{Recall@7} = 0.880$  for FEVER, which are still results consistently higher than the random sampling threshold (0.5). Finally, it is important to highlight that for the application in which the system will operate, the document corpora will have dimensions comparable with the test setting in which we operated, and hence the expected performances will be similar to those we found in our testing.

### 3.1.2 Information extractor

An important component of our architecture will be the information extractor. This process aims to extract specific information from the user input or from the generated text. The kind of information to extract greatly depends on the context of the specific application. Considering the goal of our application namely, to offer an immersive narrative and role-playing experience in a fantasy setting- extracting the right information enable the application to allow the user to freely engage with the narration through the description of their character's actions, while also modeling those actions and update the state of the adventure accordingly. After careful consideration, we selected a restricted amount of information that must be extracted either from the player's input or from the generated text: the protagonist's actions (from the user's input), if the protagonist wants to change location (again from the user's input) and if a character wants to join the protagonist (from the generated text). The extraction is performed by a linguistic model, to which we will from now on refer to as the extractor model.

**Actions and Special Actions** The first and most important type of information that we need consist in the actions that the user wants their character to perform. So we instruct our information extractor model to list all the actions that the user wants the character to perform, given the user's input. But for the purposes of our application this is not enough. We are not interested in the creation of stories in which the user is merely a passive listener. We are interested in the creation of an interactive gamified experience, in which the user takes the role of an active protagonist. This means that the story should not only react to the desires of the user, but also to the capabilities of the in-fiction character. Hence, a possible outcome of the interaction is failure. An example will make this clearer. Suppose that our user wants to impersonate a character who fits the archetype of a wizard. This character will then have great magical abilities, but will lack physical strength and fighting ability. So, if the protagonist were to find themselves up against, say, a powerful monster, and they tried to challenge it on a purely physical level, then it would only make sense for them to be defeated. If the protagonist would succeed even when, logically, this shouldn't happen, immersion would be completely lost. What follows from this reflection is that our application will need a way to objectively understand if a specific character will succeed or fail a certain action. The exact mechanics of how this happens will be explained in Section 3.2.1. What is important for the sake of this subsection is what information (other than the list of actions) we need to extract from the user's input. We need to know if one of the actions that the user wants their character to perform has an obstacle that could cause the action to fail as its target, and also which of the capabilities of the character this action requires. For this reason, we define the concept of *Special Action*, which is an object that is defined, in its JSON representation, in the following way:

```
{
  "action": string,
  "target": string,
  "information": string
}
```

where:

- "action" is the *type of action* the character is performing, that will determine which of their capabilities the character is using to perform this action;
- "target" is the subject of the action performed by the character;
- "information" contains the details regarding the performing of the action.

Let's take a simple example of user's input and the corresponding model output: **Input:** *"I want to approach the tall man with the sword, and tell him: 'Hi, traveler, what is your name?'"* **Output:**

```
{
  "action_list": ["approach the tall man with the sword","tell him: \'Hi, traveler, what is your name?\'"],
  "specific_actions": [
    {
      "action": "talking",
      "target": "Yorm the Swordmaster",
      "information": "tell him: \'Hi, traveler, what is your name?\'"
    }
  ]
}
```

The reader will notice that this example output assumes that the model has some extra context, since it is able to deduce that the "tall man with a sword" is the named character "Yorm the Swordmaster". The exact prompt and all the information it contains are available in Appendix B.

**Path taking** Another important piece of information to understand before the narration is generated is if the user intends for their character to move to a different location, e.g. to take one of the paths that lead outside the current location. To discuss how this information must be extracted, we first need to briefly introduce how the application manages locations and paths. Inside the application a single location is called a *room*. Each room has an identification number, an `int` greater than or equal to 1, and is connected to at least one other room by a *path*. So, for each room, we can associate an integer number with each of its paths, which corresponds to the id of the room it leads to. What we are interested in is exactly this id. So given the list of the paths (with the corresponding id) and the input of the user, we want to know which path (its id) the user wants the character to take, if any. If the user does not intend for the character to take any path, then we want to receive as output 0. Also in this case, an example will be informative. **Input:**

```
{
  "possible_paths": {
    "2": "A dark path, going into the oldest part of the forest",
    "3": "A path where mushrooms seem to emit some light",
    "4": "A bright path, going to the youngest part of the forest"
  }

  "action": "After carefully thinking, I decide to take the path with the strange mushrooms."
}
```

#### Output: 3

If the model returns a value different from 0, then the character is moving to another location. In this case, the application needs to change the current room, and the generated narration needs to describe the new location to the user.

**Supporting character joining the protagonist** Lastly, we must know if any character in the story joins the protagonist (controlled by the user) in their quest. In this case the input to the model is not the user's input, but the generated scene, along with a list of characters that can possibly join the protagonist. In this case the output of the model will have the following form:

```
{
  "joined": bool,
  "target": string
}
```

where "joined" is a boolean that is true if a character joined the protagonist, and "target" is the name of such character (taken from the list in input).

**Notes on the model for information extraction** At the beginning of the project, we considered using language models different from the generator model (llama-3.1-8b-instruct) for these information extraction tasks. The first candidate for these simple tasks was phi-4-mini, a small model with 3 billion parameters, that (at the time of the first steps of the project) had state-of-the-art performance that surpassed those of same size models, and also some bigger models, in many MTEB benchmarks [48, 50]. However, direct testing conducted by the researcher revealed that the model's ability to produce the desired outcomes was less than anticipated. This led to the necessity of finding a substitute model for the information extraction process. At first, an attempt was made with llama-3.2-3b-instruct, another state-of-the-art 3 billion parameter model, but also in this case the results did not meet the desired quality. Hence, the final decision was to use for these tasks the same model used in generation: llama-3.1-8b-instruct. The decision arose from a practical point of view: to conduct further research of smaller models that could perform well in those tasks would have required a considerable expenditure of time, while the model llama-3.1-8b-instruct was already available along with the literature background about its performances [32, 50] and its capabilities on the generation task had already been tested. Direct testing of this model in the three information extraction tasks showed better results compared to the performances of the 3 billion parameter models. These reasons led to the decision to adopt llama-3.1-8b-instruct as the model also for the extraction tasks. Of course this comes with a drawback: the processing time of an 8 billion parameter model is higher compared to the processing time of a 3 billion parameter model, even in simple tasks such as the ones we are considering. However, for the purposes of this research we consider the quality of the generated stories and of the experience offered by the application to be our first priority. We acknowledge that the processing time is an important aspect to consider in interactive applications, but we are also aware that an important role in the processing time is played by the hardware component, on which we have little to no control. It should be clarified that better performances on these tasks can probably be achieved by the 3 billion parameter models if properly finetuned for the tasks. This kind of finetuning procedure lies, however, outside of the scope of this research, as will be further clarified in Section 7.

### 3.1.3 Narration Generator

As previously stated, we selected as generator model for our application llama-3.1-8b-instruct, an 8 billion parameter model with state-of-the-art performance that, at the time of the architecture design, surpassed most models of comparable parameter size in most MTEB metrics [32, 50]. For the reasons presented earlier, this model ended up being also the one used for the extraction tasks.

Although an 8 billion parameter model is considered medium-small among LLMs, loading the model as is (in full precision) consumes a considerable amount of hardware resources. The full-precision version of llama-3.1-8b-instruct stores each parameter as a 32-bit floating point number, occupying a total of 32.13 GB of memory and thus requiring the same amount of RAM for inference. This solution is clearly not feasible for our research. Fortunately, quantization methods allow models of this size to run on limited hardware. These methods reduce the precision of the model parameters, leading to a substantial reduction in memory usage. Of course, this comes at a price: model performance is inevitably reduced because we lose information on the parameters. However, with a suitable choice of quantization, one may be able to run a large model using considerably less computational power without a significant performance loss. In our case, we are restricted to a maximum RAM availability of 6 GB by hardware limitations. Aware of this constraint, we chose the quantization that maximizes model performance while staying within the limit. The quantization we selected is Q5\_K\_XL:

- Q5 means that the model uses an average of 5 bits for each parameter;
- K refers to the quantization method (K-quant);
- XL indicates that this quantization is the largest in the Q5\_K family, meaning that the parameters of the most important layers are quantized with higher precision (up to 8 bits), improving model performance.

The llama-3.1-8b-instruct model with Q5\_K\_XL quantization occupies 5.7 GB of memory, which fits perfectly within our 6 GB constraint. In addition to memory savings, the quantized model allows for faster inference, which is critical for an interactive experience.

### 3.1.4 Hyperparameters choice

Regardless of the choice of the extractor and the generator model, the tasks that were previously discussed require a specific set of hyperparameters to control the model performance and adapt it to the specific task. The extraction tasks require consistent and possibly deterministic outcomes, while text generation, especially for fantasy stories, requires imaginative and varied vocabulary for the output. Within the scope of our project we are not interested in hyperparameter tuning, but rather in a suitable choice that allows the model to produce the desired results. The parameters on which we focused our attention are the following:

- **temperature:** a real value between 0 and 1. It controls the softmax probability function over the logits:

$$P_i = \frac{e^{z_i/T}}{\sum_{j=1}^V e^{z_j/T}}$$

where  $P_i$  is the probability assigned to the  $i$ th logits  $z_i$  and  $T$  is the temperature. As can be seen from the formula, for  $T \rightarrow 0$  the differences between logits are enhanced, and the logit with the higher value has a probability that tends to 1. Traditionally temperature is regarded as the parameter that controls the creativity of the model: higher values allow for more random sampling of the tokens, while a smaller value increases the probability of the more probable tokens. However, recent works [26, 58] challenge this simplistic view, showing how the temperature-creativity correlation is neither linear nor very strong, and also that the temperature is more correlated with coherence loss. On the other hand Li et al. [46] show that a correlation between temperature and creative tasks exists, and that up to a certain value of temperature the creative abilities of the LLMs increase. From the cited literature, the knowledge about the workings of the parameter and direct experience with LLM hyperparameter tuning emerges that the temperature parameter can significantly increase the creativity of the models, especially small ones, if its value is set within a certain range, but that it is not enough to guarantee the desired balance between creativity and coherence requested by a storytelling task. Moreover, it is worth noting that setting this parameter to 0 ensures that the model will always respond with the same output to a specific input;

- **top-p:** a real value between 0 and 1. It controls the cumulative probability at which a cutoff is performed. A value of 1 means that all the possible tokens will be considered in the sampling, while lowering this value discards the tokens with lower probabilities. For instance, if we have  $p=0.9$  and the tokens (with respective probabilities) are A(0.5), B(0.3), C(0.1), D(0.05), E(0.05) then we sample only over A, B and C and exclude D and E. This cutoff is performed after the computation of the probabilities, and hence it is influenced by the temperature. After the selection procedure, the probability distribution is re-computed. This method of selection, known as Nucleus Sampling, was introduced by Holtzman et al. [38] in 2020. In the cited paper it is shown how nucleus sampling consistently outperforms maximization based decoding, pure sampling and also the other main technique for truncated sampling (top-k sampling) available at the time.

- **min-p:** a real value between 0 and 1 (but usually between 0.05 and 0.1). It consists in excluding from the final sampling all the tokens whose probability is less than the  $p$  value. This technique was introduced by Nguyen et al. [51] in 2024 (the cited version is the latest, 2025). In the cited paper it is shown how min-p truncation outperforms pure sampling and other truncation techniques (among them top-p) in different tasks, including creative writing, especially at high temperature.

For the extraction tasks previously described, we do not need any creativity, rather we need consistency in the returned results. For this reason, we choose a value of temperature equal to 0.0, to ensure that the results are deterministic and avoid random sampling of less probable responses. In this way, we leverage the language analysis capabilities of the LLMs without the risk of unstable outputs. Setting the temperature to 0.0 means that we will always select the logit with the highest probability, and this makes the value of top-p irrelevant for these tasks.

For the generation task, the situation is more difficult. We need a hyperparameter setup that balances creativity and coherence. All these parameters do not act in a vacuum, but influence each other. In particular, temperature reduces the gaps between high probability and low probability tokens. Hence, let us first select a suitable value of temperature. The work of Li et al. [46] shows that the creativity of medium-sized models (6-13 billion parameters) increases slowly but steadily with temperature until values of 1.3, and then rapidly decreases with higher values. Similarly the ability of the model to follow instructions remains constant as temperature increases until 1.25, when it starts decaying. Peeperkorn et al. [58] find instead that the steeper increase in creativity (with respect to temperature) comes between 0.3 and 1.0, and that high values of temperature have a stronger negative correlation with coherence. Another important result from Li et al. comes from the study of single models in an extended temperature interval [0, 4]. In particular, one of the tested models is Meta-Llama-3-8B-Instruct, in both the full-precision version and 4-bit quantization. Both these models show high creativity performance with temperature between 0.1 and 1.0, and a steep decrease after that, and similarly high instruction following abilities until a temperature value of 1.5. This model does not correspond exactly to the one we are using, but belongs to the same family (it is the previous version of the same family) and has the same number of parameters, which means that we can probably expect similar behavior between the two models. Given these data, as well as limited qualitative testing with the model, we decided to use the temperature value of 0.9. Turning now to top-p, Holtzman et al.’s findings consistently point toward a value of top-p of 0.95, which yields the best results. It is not clear from the paper with which value of temperature this value of top-p has been found. However, insight into how this value can influence a 0.9 temperature model comes from Nguyen et al. [51]. In this paper, the results show that at temperature equal to 1.0, top-p with  $p=0.95$  outperforms pure sampling and is outperformed by top-p with  $p=0.90$ , which is in turn outperformed by min-p with  $p=0.1$ . In the light of these findings, we intend to proceed in this way: we choose a top-p value of 0.95 since, being our task a highly creative generative task, we would like to include the probability distribution up to 0.95. We however penalize extremely low probabilities with a min-p value of 0.1, which is also shown to outperform top-p sampling at similar values of temperature. To summarize, we select a temperature value of 0.9, in order to reach a wider region of the probability distribution without compromising coherence. To further reinforce coherence, we truncate the sampling at 0.95 of cumulative probability and exclude tokens with a probability score less than 0.1. These decisions are supported by knowledge of the workings of the parameters, the literature review and qualitative (even if limited) trials with the model in question (llama-3.1-8b-instruct). There are other parameters whose tuning might improve the quality of the generated story. However, a comprehensive and precise tuning of all the parameters of an LLM lies outside the scope of this project.

## 3.2. Application back-end

We now turn our attention to the specific application that will employ the aforementioned architecture. The goal of our application is to offer an immersive and interactive experience to the user, in the specific context of fantasy stories, using the characteristic interaction paradigm of TTRPGs as illustrated in Section 2.2.1. In our application the user will create a character and then embark on an adventure, in which they can interact with various elements in order to advance the story. Before designing the application, we defined a set of functionalities that the system must offer to the user.

### 3.2.1 Functionalities

#### Character creation

A core aspect of the application is that the user is not just a co-author of the story, but an active character within it. We divide the character creation into four sections, each of which specifies an important aspect of the character with respect to the story that will be generated.

**Character archetype.** An archetype is an idea, symbol, pattern, or character-type, in a story. It is any

story element that appears again and again in stories from cultures around the world and symbolizes something universal in the human experience. In the context of the character creation in our application, an archetype defines a character’s capabilities and skills, as well as the basic fantasy the character embodies. We have selected four archetypes that draw inspiration from fantasy literature and role-playing games:

- **Warrior.** *A strong and resilient fighter. Extremely skilled in combat and physical challenges, but not very good at magic or social interactions.* Probably the most common fantasy hero. Embodies the role-playing fantasy of a physically powerful skilled fighter who uses force to obtain their goal.
- **Explorer.** *A stealthy and agile character, excelling in survival and quick reflexes. Knows their way in both fighting and spellcasting<sup>1</sup>, but is not an expert in any of them.* This archetype embodies the fantasy of a treasure hunter who is always ready to face all kinds of situations with creative solutions.
- **Mage.** *A master of arcane arts, possessing vast knowledge and powerful spells. Extremely skilled in the use of magic, but not very good at fighting or physical challenges.* This archetype embodies the fantasy of wizards, sorcerers, witches or other characters that use magic and spells rather than physical abilities. It also embodies the fantasy of scholars in possession of great knowledge, and in search of more.
- **Bard.** *A charismatic performer and storyteller, skilled in music, lore, and social interactions. Can fight and use magic, but is not an expert in any of them.* This archetype embodies the fantasy of smooth talkers, performers or in general those characters that use their words and charisma to achieve their goals.

As will be clearer later, these archetypes are not only important for enabling the player to appeal to a basic character fantasy, but also to provide skills and abilities that will actively shape the unfolding of the stories.

**Character description and name.** This section includes a textual description of the character made by the user. The description can include both physical traits, character traits and backstory, and it is complemented with a name for the character. This information will be used to enable the generative model to be coherent with the nature of the character and their main traits.

**Goal of the adventure.** Like in any adventure story, the character has a main goal to achieve. In this respect, our application differs from the similar products previously analyzed (AI Dungeon and AI roguelite) which do not feature a precise goal in the adventure. Since, as explained in Section 2 we aim to reproduce a story structure, our application cannot forgo giving the user a specific goal, even at the cost of restraining the player’s freedom a little. We designed three possible goals among which the character can choose:

- *Retrieve the Mystic Crown of the Fallen King. The Mystic Crown is said to be able to grant a single wish to its bearer.* Appeals to users whose fantasy is to find ancient artifacts with great powers.
- *Defeat the evil Shadowmancer. The Shadowmancer is an evil sorcerer who plans to plunge the world into darkness.* Probably the most common trope in fantasy storytelling: defeating a great evil. Also appeals to the users who might want to live a "revenge story".
- *Find the treasure of the Fallen King. This treasure is said to be immense, and contains gems worth more than entire kingdoms.* Another very common trope in fantasy storytelling. Appeals to the users whose fantasy is to find great riches and treasures.

**Character’s motivation.** If the restriction to just three precise goals might limit the freedom of the user, the addition of a motivation partially restores this freedom. The user is invited to provide a motivation for the character to pursue the selected goal. This allows the user to tie their character directly to the main goal of the story: the goal becomes something personal that the character wants to pursue for a specific motivation.

## Interactive story generation

The other core element of the application is the interactive generation of stories. By following the paradigm highlighted in Section 2.2.1, we generate the story scene by scene according to the user input. Each scene is generated independently of the other scenes; all information needed for the generation of a scene is given to the generative model in one single prompt. This can also include information from the previous scenes. We identified five types of scenes that our system needs to generate:

- **Introductory scenes.** These scenes introduce the user to the story and location, presenting the main character and their goal and motivation (according to the choice previously made by the user).
- **Descriptive scenes.** These scenes describe to the user a location when they access it. They give a general description of the environment as well as of any important details such as paths that can be taken, characters to interact with, or obstacles to overcome.
- **Responsive scenes.** This kind of scene responds to the user input by describing the consequences of the characters’ actions.

---

<sup>1</sup>The ability to cast spells and use magic.

- **Transition scenes.** These scenes mark the transition from a main location to the next, which corresponds to the advancement of the story and to the passage to the next narrative act.
- **Ending scenes.** These scenes mark the end of a story, in which the protagonist has reached their goal, and give closure to the narration.

**Fallibility of the actions** As discussed in Section 3.1.2, an important aspect of immersion is the possibility of failing actions that are beyond the capabilities of the character. This also drives the user to adapt their interaction to their character’s abilities and the situation, and to try different approaches when one fails. RPGs usually achieve this by means of a system of characteristics and ability scores. Characteristics are numerical values that describe the intensity of basic character traits, while ability scores are numerical values that express the extent to which a character is competent at a certain class of tasks. The ability scores usually depend on the characteristic values, but can be influenced by other aspects. This will become clearer with an example.

Let us take another example from D&D. The example is illustrative, and it does not need to be understood in every detail to follow the rest of the work. A character in D&D has six characteristics (strength, dexterity, constitution, intelligence, wisdom and charisma), each of which has a value between 0 and 20. D&D also has eighteen ability scores, corresponding to various classes of tasks. For instance, let us take the ability *athletics*, which corresponds to all those tasks that require the use of physical strength. Each ability score is determined by the characteristic associated with it (for *athletics* it is strength) plus possibly a bonus depending on the character. When a character wants to attempt a task, the ability score is added to the roll of a twenty-sided die (a D20), and that determines how well the character performs the task. Let us give an example. Imagine a very strong character who has an ability score for *athletics* of +8. The character finds themselves in front of a ravine and decides to jump to the other side. The player will in this case communicate their intention to the Master, who will determine a Difficulty Class (DC), a numerical value which indicates how difficult the task is (in this case, say 18). Then the Master will ask the Player to make an ability check on *athletics*. The Player rolls a twenty-sided die and adds their character’s *athletics* ability score (+8 in this case). If the result equals or exceeds the DC (18 in our example), then the character will succeed at the task; otherwise, they will fail. The Master then narrates the outcome of the character’s action and its consequences. The key element to keep in mind after this discussion is that in this mechanism, which is common to most RPGs, the outcome of an action is decided by a precise mathematical computation, and not by either the Actor (the Player) or the Narrator (the Master).

For our application, we want to mimic this paradigm, but we want to make it simpler and more intuitive, and we want the user not to rely directly on mathematical computation to make their strategic choices. Taking these goals into account, we designed the following system of characteristics and abilities.

#### Characteristics and their descriptions

- **Strength.** *Physical power and force.*
- **Resistance.** *Ability to withstand damage and adversity.*
- **Agility.** *Skill and nimbleness in physical actions.*
- **Knowledge.** *Understanding and awareness of the world.*
- **Charisma.** *Ability to influence and inspire others.*
- **Instinct.** *Natural intuition and reflexes.*

The values and descriptions of these characteristics are known to the user, who can use them to guide what actions they want their character to perform. Each archetype previously presented starts with a specific set of values for these characteristics.

#### Abilities

- **Fighting.** Each action that consists of engaging in physical combat with a target. Its value is computed as the average of Strength, Agility and Resistance, rounded up.
- **Athletics.** Each action that consists of using physical abilities to overcome an obstacle. Its value is computed as the average of Strength and Agility, rounded up.
- **Surviving.** Each action that consists of using survival skills to overcome an obstacle or reach a goal. Its value is computed as the average of Instinct and Resistance, rounded up.
- **Tinkering.** Each action that consists of carefully interacting with an object in order to repair or modify it with precision, manual skill and finesse. Its value is computed as the average of Instinct, Agility and Knowledge, rounded up.
- **Talking.** Each action that involves interacting with a creature to reach a goal. Its value corresponds to Charisma.
- **Observing.** Each action that involves perceiving the world around using the senses. Its value corresponds to Instinct.
- **Spellcasting.** Each action that involves the use of spells or magic in general. Its value is computed as the average of Knowledge, Charisma and Instinct, rounded up.
- **Thinking.** Each action that involves using knowledge or intuition to find a solution to a problem. Its

value is computed as the average of Knowledge and Instinct, rounded up.

The user does not know the descriptions, the values or how to compute these ability scores. In fact, the user does not even know that these scores exist. The scores are computed for the character and stored in the application. Then, each time the user describes an action in their input, the information extractor will retrieve any action that belongs to any of these ability categories and return it in the form of a Special Action, as described in Section 3.1.2. Then the target of the action is searched among the obstacles that the user might face, and this value is compared with the DC of the ability for that obstacle. This will determine if the action succeeds or fails.

In the context of the application we call these obstacles Events. With Event we mean any story element that poses a challenge to the main character. This challenge can be overcome with the mechanism mentioned above: the user can describe an action with the Event as target, and the ability corresponding to the action must have a score that equals or exceeds the corresponding difficulty of the Event. Succeeding in overcoming an Event makes the character obtain a Prize. A Prize can be of two types: (1) Items (see Section 3.2.1) or (2) advancement of the story.

This mechanism ensures that the success or failure of the actions that lead the plot forward is decided not by the LLM but by the characteristics of the main character, while at the same time preventing the user from relying on purely mathematical calculations. Rather, the user must rely on their intuition to correlate the numerical values of the characteristics with the probability that a certain action might succeed or fail.

## Items and Companions

Although these are not core elements of the application, objects and supportive characters are both a staple in fantasy stories and an important element in RPGs.

By “Item” we mean, in the scope of our application, any object that can be obtained by the main character and that can help them advance in their adventure, both by increasing their characteristics and by acting as a key object to achieve a goal. For example, a sword might improve the character’s ability to fight enemies, while a key can be used to open a specific door or chest. These are not only narrative objects, but also objects that the user can manipulate in the application and that can numerically improve their character’s skills.

We now introduce the concept of “Equipment”. By Equipment we mean the set of Items that the character is actively carrying. The equipment consists of five slots: *right hand*, *left hand*, *armor*, *head* and *accessory*. Each Item is compatible only with a subset of these slots. For instance, a sword can be carried in either the right hand or the left hand slot, while a hat can only be carried in the head slot. Only an equipped Item grants a bonus to the character’s characteristics. This mimics the typical equipment mechanism of RPGs in a simplified way.

By “Companion” we mean all those characters with whom the main character can interact and who can join them on their quest, providing advantages. These characters provide hidden bonuses to the main character’s abilities, which are conveyed through the descriptions of the characters’ capabilities. For example, a character who is described as a powerful wizard and wise scholar will give bonuses to Spellcasting and Thinking.

## Feature

To complete the construction of a character, we introduce the concept of “Feature”. By Feature we mean a particular skill of a character that grants bonus to specific abilities. In the application we designed four features, one for each archetype:

- **Fighter.** *The character has fighting experience, and so has an advantage in physical combat.* Grants bonuses to Fighting and Athletics.
- **Survivalist.** *The character knows their way in the nature and can get out from difficult situations.* Grants bonuses to Surviving, Observing and Tinkering.
- **Spellcaster.** *The character is proficient in the use of magic, being able to cast better and more powerful spells.* Grants bonuses to Spellcasting and Thinking.
- **Smoothtalker.** *The character knows how to use words to convince, intimidate or deceive others.* Grants bonuses to Talking and Observing.

As in the case of companions, the bonuses are hidden to the user, and only suggested by the description of the feature.

### 3.2.2 Application design

With these functionalities in mind, we designed and implemented a demo application that could fulfill them. We will now begin by illustrating the main classes that constitute the application. The application is developed in C#, but we will describe the classes in the most language-agnostic way possible. The classes discussed in the following are described from a design point of view. In the actual implementation of the application, some differences could be found with respect to the following descriptions. For instance, some classes or some

dictionaries are actually implemented as **struct**. More precisely, what we describe abstractly as a dictionary (e.g., mapping attribute names to integer values) might be in the code a struct with fixed fields (for example, a struct containing ten integer attributes instead of a `Dictionary<string, int>`). Although implemented differently, these solutions are conceptually equivalent to the ones described in this section.

**Character** This class is used to deserialize and temporarily store the character created by the user, sent as a JSON through a POST API call from the front-end. It contains the attributes listed in Table 3.

Table 3: Attributes of the **Character** class.

| Attribute       | Type                       | Description                                                                                                                                                |
|-----------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name            | string                     | The character's name.                                                                                                                                      |
| description     | string                     | A textual description of the character (physical traits, personality, backstory).                                                                          |
| archetype       | string                     | The chosen archetype (Warrior, Explorer, Mage, or Bard).                                                                                                   |
| characteristics | Dictionary(string, int)    | The six numerical characteristics (Strength, Resistance, Agility, Knowledge, Charisma, Instinct).                                                          |
| equipment       | Dictionary(string, string) | The set of Items currently carried, organized into slots. The dictionary only contains the name of the Items, as a result of the de-serialization process. |
| feature         | string                     | A special ability or trait associated with the archetype. The attribute only contains the name of the feature                                              |

**Item** This class implements an Item as described in Section 3.2.1. Its attributes are listed in Table 4.

Table 4: Attributes of the **Item** class.

| Attribute       | Type         | Description                                                                                          |
|-----------------|--------------|------------------------------------------------------------------------------------------------------|
| name            | string       | The item's name (e.g., "Steel Sword", "Mystic Hat").                                                 |
| description     | string       | A textual description of the item.                                                                   |
| compatibilities | List<string> | A list of strings indicating which equipment slots the item can occupy (e.g., "right hand", "head"). |
| characteristic  | string       | The characteristic (e.g., Strength, Agility) that is improved when the item is equipped.             |
| bonus           | int          | An integer representing the amount of bonus granted to the specified characteristic.                 |

**Companion** This class implements a Companion as described in Section 3.2.1. Its attributes are listed in Table 5.

Table 5: Attributes of the **Companion** class.

| Attribute           | Type                    | Description                                                                                                                                  |
|---------------------|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| name                | string                  | The name of the companion.                                                                                                                   |
| description         | string                  | A textual description of the companion's appearance.                                                                                         |
| ability_description | string                  | A textual description of the companion's abilities. It is intended to be shown to the user as a hint about the bonuses the companion offers. |
| bonuses             | Dictionary<string, int> | The set of abilities improved by the companion with their relative bonuses.                                                                  |

**Feature** This class implements a Feature as described in Section 3.2.1. Its attributes are listed in Table 6.

Table 6: Attributes of the **Feature** class.

| Attribute          | Type                    | Description                                                         |
|--------------------|-------------------------|---------------------------------------------------------------------|
| <b>name</b>        | string                  | The name of the feature (e.g., Fighter, Survivalist).               |
| <b>description</b> | string                  | A textual description of the feature.                               |
| <b>bonuses</b>     | Dictionary(string, int) | The set of abilities improved by the feature with relative bonuses. |

**Event** This class implements an Event as described in Section 3.2.1. Its attributes are listed in Table 7.

Table 7: Attributes of the **Event** class.

| Attribute              | Type                                        | Description                                                                                                                                                                                         |
|------------------------|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>name</b>            | string                                      | The name of the event.                                                                                                                                                                              |
| <b>description</b>     | string                                      | A textual description of the event.                                                                                                                                                                 |
| <b>difficulties</b>    | Dictionary<string, (int dc, Prizes prizes)> | Keys are ability names (e.g., Fighting). Value is a composite: integer dc (difficulty class, see Section 3.2.1) and <b>Prizes</b> (custom type, either List<Item> or string).                       |
| <b>block_path</b>      | bool                                        | Whether this event blocks the protagonist from advancing or changing location.                                                                                                                      |
| <b>bonus_objects</b>   | Dictionary<string, string>                  | Keys are ability names (e.g., Fighting). Value is an Item name. If the Item is in <b>inventory</b> or <b>equipment</b> , the ability gains a bonus. Models task facilitation (e.g., key for chest). |
| <b>following_event</b> | string                                      | Name of the event that immediately follows after completion. Null or empty if none.                                                                                                                 |

**Room** This class implements a Room. In the context of this application, a Room represents a single location of the story. Its attributes are listed in Table 8. This class has one method, `pathBlocked()`, which returns true if at least one of the events in the room has the attribute `block_path` equal to true; otherwise, it returns false.

Table 8: Attributes of the **Room** class.

| Attribute          | Type            | Description                                                                     |
|--------------------|-----------------|---------------------------------------------------------------------------------|
| <b>id</b>          | int             | The id of the Room, from 0 to 999. The id is unique inside an Act.              |
| <b>description</b> | string          | A textual description of the location represented by the room.                  |
| <b>companions</b>  | List<Companion> | The list of Companions who can be encountered in the Room. Can be empty.        |
| <b>events</b>      | List<Event>     | The list of Events that are interactable in the room. Can be empty.             |
| <b>paths</b>       | List<string>    | The list of all the paths leading outside the location represented by the room. |
| <b>connections</b> | List<int>       | The list of all the rooms that are reachable from this one.                     |

**Act** This class implements a narrative act. Its attributes are listed in Table 9.

Table 9: Attributes of the **Act** class.

| Attribute                  | Type            | Description                                                                                      |
|----------------------------|-----------------|--------------------------------------------------------------------------------------------------|
| <b>number</b>              | int             | The number of the act, from 1 to 3 in our setting.                                               |
| <b>location</b>            | string          | The main location in which the act takes place.                                                  |
| <b>companions</b>          | List<Companion> | The list of Companions who can be encountered in the act. Can be empty.                          |
| <b>introductory_prompt</b> | string          | A brief description of the introduction of the act, used to generate the introductory narration. |
| <b>rooms</b>               | List<Room>      | The list of the rooms in the act.                                                                |

**Room Manager** This static class manages the rooms that make up the adventure. The attributes of the class are listed in Table 10. The main methods of the class are listed in Table 11.

Table 10: Attributes of the Room Manager class.

| Attribute   | Type                  | Description                                                                                                                |
|-------------|-----------------------|----------------------------------------------------------------------------------------------------------------------------|
| currentRoom | Room                  | The room corresponding to the current location of the story.                                                               |
| lastRoomId  | int                   | The id of the last room visited. When entering a new room, it is used to understand which path the user is using to enter. |
| rooms       | Dictionary<int, Room> | The set of rooms currently loaded, paired with the corresponding id.                                                       |
| ROOM_LIMIT  | int                   | A read-only constant that sets the hypothetical maximum value of the id of the rooms; in this case, it is set to 999.      |

Table 11: Methods of the Room Manager class.

| Method          | Parameters => Return Type | Description                                                                                                                                                                                                                       |
|-----------------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| InitializeRooms | () => void                | Loads the rooms of the current Act and sets the current room to the room with id equal to 1. For the third act, the starting room depends on the goal selected by the user.                                                       |
| MoveRoom        | (int id) => bool          | Sets lastRoomId equal to the current room id and changes currentRoom to the room with id equal to the parameter id. If id is greater than ROOM_LIMIT, then it increases the current narrative Act by 1 and calls InitializeRooms. |

**World State Info** This static class contains many different pieces of information on the current state of the application, the story and, mainly, the user’s character. The attributes of the class are listed in Table 12. The main methods of the class are listed in Table 13.

Table 12: Attributes of the World State Info class.

| Attribute       | Type                     | Description                                                                                                                  |
|-----------------|--------------------------|------------------------------------------------------------------------------------------------------------------------------|
| character       | Character                | The character created by the user.                                                                                           |
| ability_scores  | Dictionary<string, int>  | The values of the abilities of the character as defined in Section 3.2.1.                                                    |
| character_equip | Dictionary<string, Item> | The set of Items currently carried by the character, organized into slots. This dictionary contains the actual Item objects. |
| inventory       | List<Item>               | The Items possessed by the character but not equipped.                                                                       |
| features        | List<Feature>            | The features of the character.                                                                                               |
| party           | List<Companion>          | The list of companions that accompany the character.                                                                         |
| objective       | string                   | The goal of the character as discussed in Section 3.2.1.                                                                     |
| motivation      | string                   | The character’s motivation as discussed in Section 3.2.1.                                                                    |
| act             | int                      | The current narrative act.                                                                                                   |

Table 13: Methods of the World State Info class.

| Method                  | Parameters => Return Type            | Description                                                                                                                                                                                                                                                         |
|-------------------------|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>addCharacter</b>     | (Character c) => void                | Takes a Character as input, sets it as the <b>character</b> attribute, converts the strings for features and items into the corresponding objects, computes the current characteristics given the bonuses from the items, and calls the <b>computeStats</b> method. |
| <b>computeStats</b>     | () => void                           | Computes the ability scores of the character starting from its characteristics, equipped Items and Companions.                                                                                                                                                      |
| <b>getCharacterInfo</b> | () => Dictionary<string, object>     | Returns a dictionary containing information about the character ready to be sent as a response to an API call.                                                                                                                                                      |
| <b>addPartyMember</b>   | (Companion cm) => Companion          | Adds <b>cm</b> to the <b>party</b> attribute and removes it from the current Room.                                                                                                                                                                                  |
| <b>addItem</b>          | (List<Item> items) => void           | Adds the elements of <b>items</b> to the <b>inventory</b> .                                                                                                                                                                                                         |
| <b>switchEquipments</b> | (Item item, string position) => void | Inserts <b>item</b> into <b>character_equip</b> in the slot indicated by <b>position</b> . If the slot already contains an Item, that Item is transferred to <b>inventory</b> . Then recomputes the characteristics and calls the <b>computeStats</b> method.       |

**Special Action** This class implements a Special Action as described in Section 3.1.2. Its attributes are listed in Table 14.

Table 14: Attributes of the Special Action class.

| Attribute          | Type         | Description                                                                                                 |
|--------------------|--------------|-------------------------------------------------------------------------------------------------------------|
| <b>action</b>      | string       | The “type of action” to be performed, which corresponds to one of the abilities described in Section 3.2.1. |
| <b>target</b>      | string       | The name of the Companion or Event which is the target of the action.                                       |
| <b>information</b> | List<string> | Additional information about the action.                                                                    |

**Formatted Action** This class is used to enrich a Special Action with additional attributes to make it usable for building the prompt for story generation. Its attributes are listed in Table 15. This class has only one method, **FormatActions**, which takes as input a List of Special Actions and outputs a List of Formatted Actions. For each Special Action in the input, the method finds the corresponding ability score of the character and the corresponding target event. It then compares the event’s difficulty for that ability with the ability score and assesses the success or failure of the action. In doing so, it checks whether there are any items in the character’s equipment or inventory that grant them bonuses (as discussed in Section 3.2.2). Then it checks the prizes for the successful completion of the event and fills the **prizes** attribute accordingly. If the target is a companion, the action automatically succeeds, but no prize is registered. Ideally, the application should also deal with the fallibility of actions toward a companion, but, given time and resource constraints, for the sake of the demo we decided to ignore this aspect.

Table 15: Attributes of the Formatted Action class.

| Attribute          | Type         | Description                                                                                                                                                                          |
|--------------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>action</b>      | string       | The “type of action” to be performed, which corresponds to one of the abilities described in Section 3.2.1.                                                                          |
| <b>target</b>      | string       | The name of the Companion or Event which is the target of the action.                                                                                                                |
| <b>information</b> | List<string> | Additional information about the action.                                                                                                                                             |
| <b>success</b>     | bool         | A boolean that describes whether the action has succeeded or failed, in the sense discussed in Section 3.2.1.                                                                        |
| <b>prizes</b>      | string       | A concatenated string listing the prizes of an action. If the prizes are Items, it lists their names. If the prize is a narrative string, this attribute corresponds to that string. |

**App Data** This static class reads, stores, and grants access to the data of the application. The data on which the application works (Events, Companions, Rooms, etc.) are stored in a JSON file. This allows the data to be easily tweaked, even without a deep knowledge of the app, thus making it possible to change or adjust the adventure that the user will experience. The attributes of the class are listed in Table 16. The main methods of the class are listed in Table 17.

Table 16: Attributes of the App Data class.

| Attribute  | Type            | Description                                       |
|------------|-----------------|---------------------------------------------------|
| DataPath   | string          | The path of the JSON file that contains the data. |
| items      | List<Item>      | A list of all the Items available.                |
| features   | List<Feature>   | A list of all the Features available.             |
| events     | List<Event>     | A list of all the Events available.               |
| companions | List<Companion> | A list of all the Companions available.           |
| acts       | List<Act>       | The acts that compose the story.                  |

Table 17: Methods of the App Data class.

| Method             | Parameters => Return Type                  | Description                                                                                                                                                                                                                                 |
|--------------------|--------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Init               | () => string                               | Deserializes JSON data using temporary structs, fills <code>items</code> , <code>features</code> , <code>companions</code> . Calls <code>DeserializeEvents</code> and <code>DeserializeActs</code> . Returns "success" or an error message. |
| DeserializeEvents  | (Data data) => void                        | Composes events from temporary data. Finds Items by name inside the <code>prizes</code> attribute.                                                                                                                                          |
| DeserializeActs    | (Data data) => void                        | Composes Acts from temporary data. Constructs Room objects and populates their Events and Companions by name lookup.                                                                                                                        |
| getItemByName      | (string name) => Item                      | Returns the Item with matching <code>name</code> , or null if not found.                                                                                                                                                                    |
| getFeatureByName   | (string name) => Feature                   | Returns the Feature with matching <code>name</code> , or null if not found.                                                                                                                                                                 |
| getEventByName     | (string name) => Event                     | Returns the Event with matching <code>name</code> , or null if not found.                                                                                                                                                                   |
| getCompanionByName | (string name, bool tolerance) => Companion | Returns the Companion with matching <code>name</code> , or null if not found. If <code>tolerance</code> is true, handles misspellings and partial names (e.g., from LLM output).                                                            |
| getActRooms        | (int actId) => List<Room>                  | Returns the rooms of the Act with matching <code>number</code> .                                                                                                                                                                            |
| getActPrompt       | (int actId) => string                      | Returns the <code>introductory_prompt</code> of the Act with matching <code>number</code> .                                                                                                                                                 |
| getActLocation     | (int actId) => string                      | Returns the <code>location</code> of the Act with matching <code>number</code> .                                                                                                                                                            |

**Narrator** This static class contains the methods to compose the various prompts needed for the functions of the architecture discussed in Section 3.1. The attributes of the class are listed in Table 18. The main methods of the class are listed in Table 19.

Table 18: Attributes of the Narrator class.

| Attribute   | Type   | Description                                          |
|-------------|--------|------------------------------------------------------|
| _lastScene  | string | The last scene of the story.                         |
| _storySoFar | string | A summary of the story in the current narrative act. |
| _totalStory | string | A summary of the total story, divided by acts.       |

Table 19: Methods of the `Narrator` class.

| Method                               | Parameters => Return Type                                                                                | Description                                                                                                                                            |
|--------------------------------------|----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>BuildIntroductionPrompt</code> | () => string                                                                                             | Constructs the prompt for an Introductory Scene (see Section 3.2.1).                                                                                   |
| <code>BuildBasicPrompt</code>        | (List<string> actions, List<FormattedAction> formattedActions, List<string> context, int path) => string | Constructs the prompt for a Responsive or Descriptive scene (see Section 3.2.1). Recognizes act changes and calls <code>BuildTransitionPrompt</code> . |
| <code>BuildTransitionPrompt</code>   | (List<string> actions) => string                                                                         | Constructs the prompt for a Transition Scene (see Section 3.2.1). Recognizes story end and calls <code>BuildEndingPrompt</code> .                      |
| <code>BuildEndingPrompt</code>       | (List<string> actions) => string                                                                         | Constructs the prompt for an Ending Scene (see Section 3.2.1).                                                                                         |
| <code>UpdateWithSummaryAsync</code>  | async (string newNarration) => Item                                                                      | Generates a summary of the last scene and updates <code>_lastScene</code> and <code>_storySoFar</code> .                                               |
| <code>UpdateCompleteStory</code>     | () => void                                                                                               | Updates <code>_totalStory</code> with <code>_storySoFar</code> and clears it. Called at the end of each act.                                           |
| <code>BuildHasJoinedRequest</code>   | (string generatedNarration, List<Companion> currentCompanions) => string                                 | Constructs the prompt to check if any companions joined the protagonist (see Section 3.1.2).                                                           |

**API Caller** This static class contains the methods to send requests to the API of the architecture in order to access to the functionalities discussed in Section 3.1. The attributes of the class are listed in Table 20. The main methods of the class are listed in Table 21.

Table 20: Attributes of the `API Caller` class.

| Attribute               | Type                       | Description                                                   |
|-------------------------|----------------------------|---------------------------------------------------------------|
| <code>APIBaseURL</code> | string                     | A constant containing the URL where to send the requests.     |
| <code>roots</code>      | Dictionary<string, string> | The roots to reach the various functions of the architecture. |
| <code>client</code>     | HttpClient                 | The client used to send the requests to the API.              |

Table 21: Methods of the API Caller class.

| Method          | Parameters => Return Type                                       | Description                                                                                                                                                                                                                   |
|-----------------|-----------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Init            | () => void                                                      | Setup the client.                                                                                                                                                                                                             |
| CallAPI<T>      | (string method, string endpoint, string body = null) => Task<T> | Send a requests of the type specified in method to the root corresponding to <code>roots[endpoint]</code> . Uses the body in case of a POST request, if needed. Returns a task of type T.                                     |
| Search          | (string query) => Task<SearchResponse>                          | Sends a POST request to the retrieval endpoint. Query contains the query terms for the retrieval system. SearchResponse is a struct used for the de-serialization of the response.                                            |
| ExtractInfo     | (string query) => Task<InfoExtractionResponse>                  | Sends a POST request to the info extraction endpoint. Query contains the prompt for the extraction of actions and special actions. InfoExtractionResponse is a struct used for the de-serialization of the response.          |
| FindPath        | (string query) => Task<PathfindingResponse>                     | Sends a POST request to the path taking endpoint. Query contains the prompt for the extraction of the path taken by the user. PathfindingResponse is a struct used for the de-serialization of the response.                  |
| Generate        | (string query) => Task<GeneratorResponse>                       | Sends a POST request to the narration generation endpoint. Query contains the prompt for the generation of the current scene. GeneratorResponse is a struct used for the de-serialization of the response.                    |
| GenerateSummary | (string query) => Task<string>                                  | Sends a POST request to the summary generation endpoint. Query contains the prompt for the generation of the summary of the current scene.                                                                                    |
| HasJoined       | (string query) => Task<HasJoinedResponse>                       | Sends a POST request to the info extraction endpoint. Query contains the prompt for the extraction of wether a character joined the protagonist. HasJoinedResponse is a struct used for the de-serialization of the response. |
| RebuildIndexes  | () => void                                                      | Sends a POST request to the rebuild index endpoint. Cause the rebuilding of indexes of the documents used by the retrieval system.                                                                                            |

**Program** This is not a class, but rather the entry point of the API of the back-end. It is responsible for building the host, registering services, and defining the endpoint mappings. The API endpoints exposed and the tasks they perform are listed in Table 22.

Table 22: API endpoints exposed by the back-end.

| Endpoint               | Parameters                            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GET /init              | ()                                    | Initializes the application, calling various methods from other classes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| POST /character        | (Character character)                 | Saves the character in the <code>WorldStateInfo</code> class, calling the method <code>addCharacter</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| GET /character         | ()                                    | Returns the character stored in <code>WorldStateInfo.character</code> by calling the method <code>WorldStateInfo.getCharacterInfo</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| POST /objective        | (string objective, string motivation) | Sets <code>WorldStateInfo.objective</code> and <code>WorldStateInfo.motivation</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| POST /start            | ()                                    | Starts the story by generating the Introductory Scene.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| POST /first_scene      | ()                                    | Generates the first Descriptive scene to start the interaction. It is called after the Introductory scene and every Transition scene.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| POST /action           | (ActionRequest action = null)         | Receives the user input (the description of the actions of the character) and performs all the operations needed to generate the Responsive scene. It calls the methods <code>APICaller.Search</code> , <code>APICaller.ExtractInfo</code> and <code>APICaller.FindPath</code> , and uses the information obtained by these methods to build the prompt for the scene generation, by calling the <code>Narrator.BuildBasicPrompt</code> method. Then it uses this prompt to generate the scene with <code>APICaller.Generate</code> , and uses the generated scene as a parameter for the method <code>Narrator.BuildHasJoinedRequest</code> . It then uses the return value of this last method as a parameter for <code>APICaller.HasJoined</code> . If this method returns a valid Companion, it gets added to the main character's party through the method <code>WorldStateInfo.addPartyMember</code> . This method also checks if the story has ended, and calls all the necessary methods. Finally, it returns the generated text along with some useful information on the state of the application (namely, if the narrative act has changed, if the story has ended and which Companion, if any, has joined the main character). |
| POST /equipment/switch | (SwitchEquipmentRequest request)      | Triggers the insertion of an Item into the character's equipment in a specific slot. <code>SwitchEquipmentRequest</code> is a struct which contains the attributes <code>ItemName</code> and <code>Position</code> . These are used as parameters of the method <code>WorldStateInfo.switchEquipment</code> . Finally, this endpoint returns the character information obtained from the method <code>WorldStateInfo.getCharacterInfo</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| GET /inventory         | ()                                    | Returns the inventory of the character.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

### 3.3. Application front-end

Although it is not the main focus of our work, a suitable front-end for the application plays an important role in offering an immersive and enjoyable experience. One of the advantages of our architecture is precisely that it allows the user to interact with the story not only by describing their actions, but also through other interactions that have an actual impact on the story. We are specifically referring to the ability of the user to visualize the Items obtained during the adventure and equip them in order to improve the character’s characteristics. This enhances both engagement and immersion, since the user feels that their actions have concrete, tangible rewards which impact the character’s progression and improvement in a meaningful way.

To translate these interactions into a smooth and intuitive experience, we designed a front-end that prioritizes clarity of feedback and ease of use, ensuring that players can focus on the narrative without being distracted by the underlying complexity.

#### 3.3.1 UI/UX design

In the following, we will analyze each screen of the application and discuss the motivations behind the main design choices. Before that, we briefly discuss some implementation details.

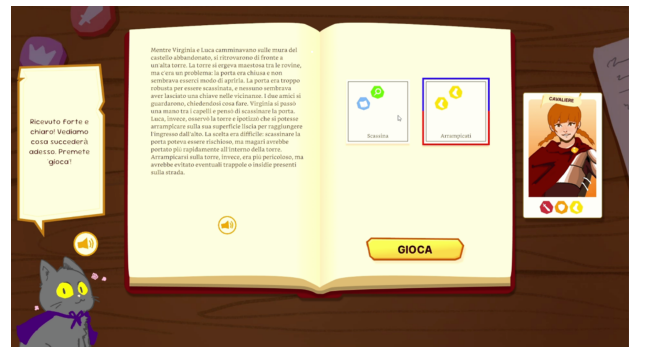
**Notes on implementation.** We will not delve into the implementation details of the front-end, since they are of little interest for the scope of this work: the front-end could be re-implemented with different technologies (e.g., Unity) without affecting the core architectural choices or the back-end logic. The specific implementation does not significantly impact the overall functioning of the application. We chose to implement the front-end as a web application using JavaScript as the main programming language, with React as the UI library and Vite as the build tool. This choice was motivated by ease of implementation, the extensive ecosystem and community support, and the flexibility and portability of the resulting solution. Some notable libraries used in the implementation are `@dnd-kit/core`, which manages drag-and-drop of certain UI elements (namely, the Items), and `zustand`, which provides a lightweight state management solution for temporary application data.

**Starting screen.** This is the first screen with which the user interacts during the experience. It is shown in Figure 2a. The main element of this screen, which will be a recurring element throughout the whole UI, is the Book. The Book is the central UI element on which most of the user interaction occurs. The idea of reproducing a book as the main interactive element of the UI comes mainly from Valcamonica et al. [76], in which the authors also operate in the context of interactive fantasy storytelling with LLMs, albeit with different goals. In general, the UI of their application has been a major inspiration for the design of our own. We have dedicated less time and effort to the quality of the visual assets of our UI, since in our case it is not the main focus of the work.

The book is divided into two main parts: the left page and the right page. Each page hosts different UI elements. In the starting screen, the left page welcomes the user and briefly presents the application, and allows them to start the experience. The right page instead shows a brief paragraph that introduces the user to the narrative setting and to the incipit of the story they will experience.



(a) Starting screen of the application



(b) UI of the application from Valcamonica et al.

Figure 2: Comparison between our UI and the one from Valcamonica et al.

**Archetype selection screen.** In this screen, the user can select the archetype of their character (see Section 3.2.1). On the left page, the four archetypes are selectable by clicking on the corresponding button. When an archetype is selected, its information is displayed on the right page. In particular, a panel shows the

starting characteristics of the archetype, its starting Equipment and the associated Feature (see Section 3.2.1). It also offers a brief description of the abilities of a character with the selected archetype. The archetype selection screen is shown in Figure 3.



Figure 3: Archetype selection screen.

**Character information screen.** In this screen, the user can choose a name and provide a description for their character (see Section 3.2.1). On the left page, the same panel as in the archetype selection screen appears, showing the character's parametric features. In this case, however, the parchment that in the archetype selection screen displayed the archetype name becomes an input text field in which the character's name can be written. On the right page, another input text field allows the user to enter the character description. The character information screen is shown in Figure 4.

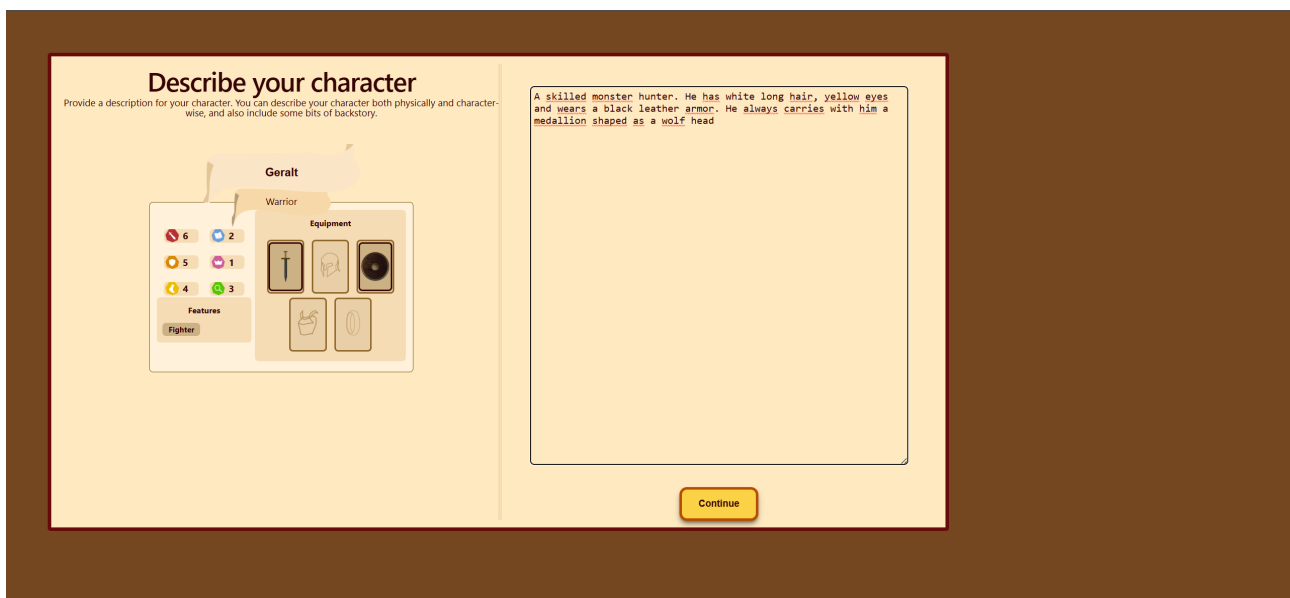


Figure 4: Character information screen.

**Goal and motivation screen.** In this screen, the user can select the goal of the adventure (among three available options) and specify the motivation of the character (see Section 3.2.1). On the left page, three buttons allow for the selection of the goal. On the right page, an input text field allows the user to enter the character's motivation. The goal and motivation screen is shown in Figure 5.

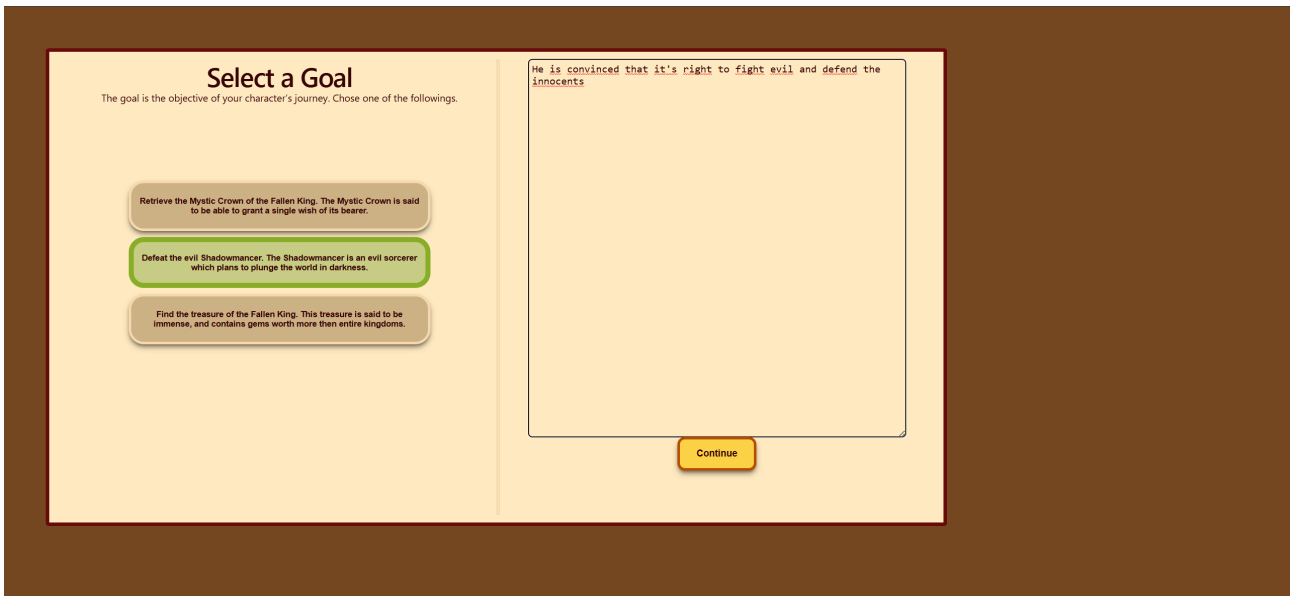


Figure 5: Goal and motivation screen.

**Narrative screen.** This is the main screen of the application, which presents the generated text to the user and allows them to specify their character's actions. On the left page, the screen shows the current narration. On the right page, an input text field allows the user to enter the character's action. Moreover, this page introduces the other main element of the UI: the "Character sheet". This is a panel, similar to the ones seen in the archetype selection screen and in the character information screen, but it has a vertical layout. It sums up all the main (non-narrative) information about the character. It shows the characteristics, the Equipment and the Feature of the character. In addition, it shows two additional pieces of information:

- the character's inventory, which contains all the character's non-equipped Items. From this section, the user can drag the Items and drop it onto one of the equipment slots to equip the corresponding Item. This mechanism automatically manages the compatibilities of the Items (see Section 3.2.1) and performs the corresponding API call (see Section 3.2.2);
- the character's party, which shows all the companions who follow the main character in their adventure (see Section 3.2.1).

The narrative screen is shown in Figure 6.

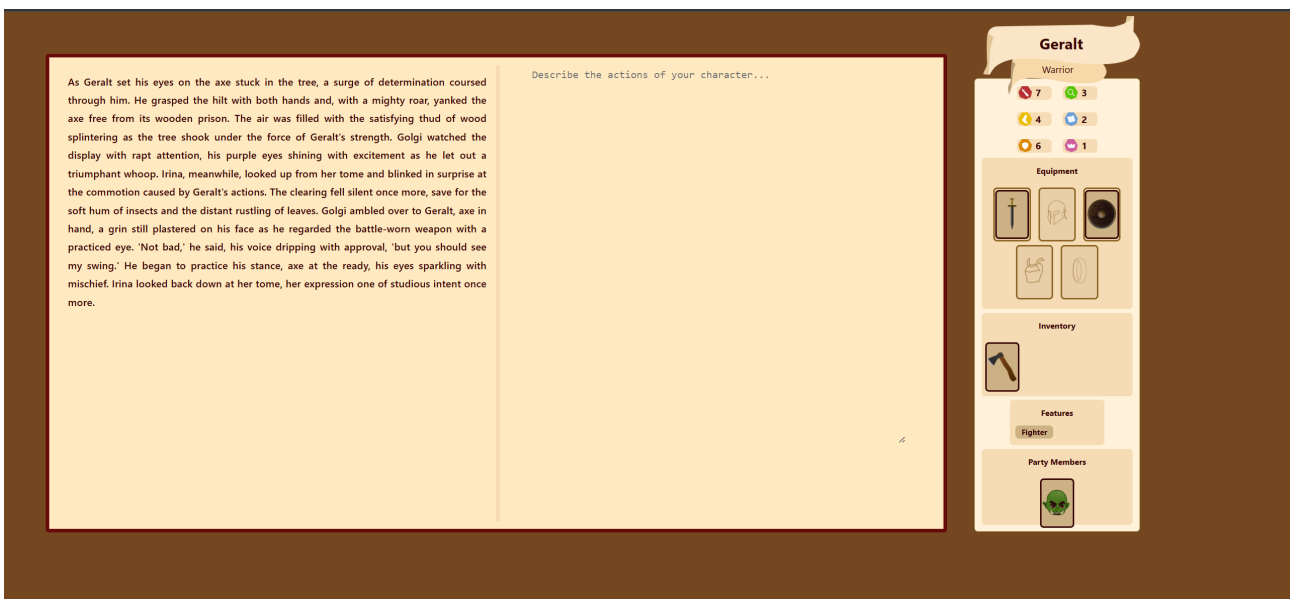


Figure 6: Narrative screen.

**Additional Notes on the UI.** To prevent confusion, each UI element corresponding to an application component (Characteristics, Items, Companions, and Features) displays a tooltip when hovered with the mouse.

The tooltip contains a description of the element and its impact on the story. For Characteristics and Features, these are the descriptions in italics from Section 3.2.1. For Companions, the tooltips focus mainly on their capabilities, as discussed in Section 3.2.1, while physical or personality descriptions are left to the narrative. For Items, the tooltip provides not only a description, but also information about compatible equipment slots and the bonus they grant to characteristics (see Section 3.2.1).

### 3.4. Additional Notes on the application and the architecture

In this subsection, we will address some aspects of the application that have not been covered previously.

#### 3.4.1 Relation between application and architecture

In Section 3, we discussed how the architecture and the specific application that uses it are separate components. While this holds true in theory, in practice there are some overlaps between the architecture and the application.

First, it is important to stress that the architecture is not completely application-agnostic: it has been designed with a specific goal in mind, namely the generation of interactive and immersive fantasy stories with role-playing game elements. This means that, to use the architecture as is, an application must be at least similar to the one actually developed in this work. However, the architecture can be easily refactored for other purposes without the need to heavily modify the code. By only tweaking the system prompts for each component, it is possible to modify the genre of the produced story (thriller, action, horror), change the desired linguistic register, and alter the structure of the generated scenes and the overall narrative arc, as well as the specific types of information extracted by the information extractor. Moreover, since the architecture is based on API calls, a client application can freely call only the necessary functions, in any order. Additionally, thanks to its modularity, new functions can be added to the application with relative ease, without requiring a deep understanding of the existing ones.

Secondly, as mentioned, there are some overlaps between the architecture and the application, meaning that some components theoretically belonging to the application are implemented as part of the architecture, and vice versa. The two most notable examples are the documents used for the retrieval system and the story structure file.

The documents used by the retrieval system are stored as a JSON file in the `./data` folder of the architecture code. Clearly, these documents do not belong to the architecture, since they contain information about the narrative setting specified by the application. However, they were stored directly within the architecture to simplify the implementation. Nonetheless, the architecture API offers dedicated calls to load new documents and recompute their indexes.

The story structure file (again a JSON file) is stored in the `./Resources` folder of the application. However, one could argue that this file should actually be a product of the architecture, since it represents the story outline, which would normally be provided as plain text to the LLM. In fact, adopting a black-box view of the entire process, the input to the simple LLM and to the architecture should be the same, namely the user input for each scene and the story outline. This is a valid objection, which will be addressed in the next section.

#### 3.4.2 The author perspective

Up to this point we have focused as target users the 'players', the people which experience the interactive and immersive story generated by the application. However, another potential class of users can be interested in this application: we will call them the 'authors'. We have discussed in Section 2.2.1 how the Master can assume both the role of author and narrator of the story that the Players experience. In this work we focused on the second role: our application does not author the story, does not generate the *histoire*, but only the *discourse* (see Section 2.1). We are hence considering the story author as an external agent, with respect to the system. However, this means that we can expect from this agent the same input to the system that would be given to a simple LLM: a textual story outline. This text can be very detailed, including a large amount of information regarding the story, but it would still be a purely textual input, written in natural language. In our application, however, we use as story outline a JSON file, containing all the information described in detail in Section 3.2. One could argue that a confrontation between the simple LLM, which uses the natural language story outline, and the architecture, which employs the formatted story outline, cannot be performed, since there is a larger amount of information on the second case. The same objection could be moved about the use of the retrieval system, which provides more information to the model, with respect to the non architecture case. However in this case the extra information is completely a product of the architecture: even if the author would supply the documents about the narrative setting to an application which doesn't employ the architecture, there is no way

to provide those information to the generative model (except for providing to the model the entire document corpus, which is not feasible). In the case of the story outline, however, there is not any part of the architecture (currently) which produces the formatted story outline starting from the one in natural language. In this sense, one could object that there is an extra human effort behind the story generated with the architecture. This is however not a limit of the architecture itself, rather than the scope of this work. In fact, the architecture could be easily extended to produce, given in input a story outline in natural language, the corresponding formatted story outline. This would nonetheless require a model large enough to perform such a complex task satisfactorily (or alternatively a finetuned model) and careful prompt engineering. We did not engage in the implementation of this function for the following reasons:

- **Scope of the research.** We are interested mainly in the interactive generation of the story in response to the user inputs. The adherence to the story outline is only one of the criteria we evaluate, and is always evaluated against the natural language outline. Hence, the extra information which might arise from a human compiled formatted outline have little influence in this sense, since they would not be counted as adherent to the outline. The real difference would only be given by how the application and the architecture manage the information of the outline.
- **Limited resources.** As previously stated, this translation function would require a non-negligible amount of work, time and computational resources. Given the limited resources at our disposal and the marginality of this passage to our scope, we decided to process manually the story outline, especially considering that our study focus specifically on a single story outline.

Figure 7 shows how the current architecture could be extended with a component which performs the conversion from a natural language outline to a formatted one.

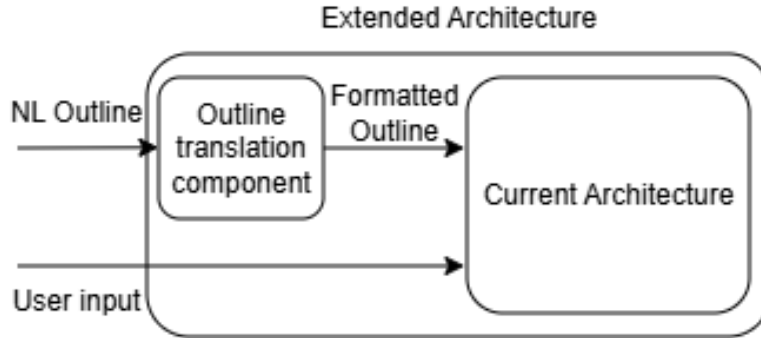


Figure 7: Extended Architecture.

## 4. Testing of the solution

In this section, we discuss the testing of the solution. We tested our solution by means of human testing and evaluation, while collecting data about the experiences to perform some statistical evaluation on them. We will begin by addressing the limitations of the performed testing, so that the reader can better understand the motivations behind some specific choices that were made. Later, we will discuss the general experimental setup and the specific experiences of the users with the system.

### 4.1. Limitations of the testing

We encountered various limitations and difficulties in our test campaign for the system. Most of these difficulties are due to the limited resources at our disposal, while others are due to the very nature of the system. The main limitation was the need to run the entire application on a single device. It was not possible to set up a server capable of hosting the application and serving multiple users at the same time. The entire application (from the front-end to the models) was hosted and executed on a single device, allowing only one test to be performed at a time. Moreover, this accentuated an intrinsic problem of the application: the processing time. The application is implemented in such a way that it can run parallel calls when possible. For example, the operations of extracting actions from the user input and path taking (see Section 3.1) can be performed in parallel and later used to produce the narration. However, this is not possible in the local setup: the overhead of OS RAM consumption and the limited hardware allow only a single model to run on LM Studio. Also, the device used does not possess a modern high-performance GPU, so even a relatively small model such as Llama-3.1 takes a considerable amount of time for computation. The combination of these two factors (the one-device solution and the slow generation time) considerably slowed the data collection process. This problem was aggravated by

the limited time available for testing.

Another problem, directly linked to the nature of the architecture and the experience it offers, is the long duration of a single test. To evaluate the effectiveness of the architecture, the story needed to be long and structured enough to meaningfully challenge the context maintenance and long-term coherence capabilities of the LLM. However, considering the long duration of a single interaction, this was not possible to a satisfactory extent. In trying to find a solution that balanced the excessive time consumption of a single test with the need to evaluate long stories, we crafted a story outline consisting of a total of four rooms (one for the first Act, two for the second, and one for the third). The details of this story outline, and its specific limitations, will be addressed in the Appendix C. For now, we will focus on the impact of this choice on the test duration. The tests produced stories with scenes ranging between 18 and 30 scenes, with durations between 1 hour 42 minutes and 2 hours 41 minutes. While this number of scenes is likely insufficient to fully assess the qualitative benefits of the architecture on narrative coherence and variety (which typically emerge beyond 20 scenes), it is sufficient to evaluate the quantitative impact—such as context length reduction and generation time—which are measurable from the first interactions. At the same time, the duration of a single test is extremely long in the context of our testing campaign. This duration is not necessarily problematic for the user experience itself—immersive narrative sessions of similar length are common in role-playing games—but it became a critical issue for testing, as it made it difficult to recruit volunteers and to run multiple tests within the limited time available.

Another problem, which is aggravated by the long duration of a test, is finding users to test the system. We did not have the resources to recruit paid testers. All testers of the application were volunteers. As the reader will easily appreciate, it is difficult to find people willing to participate in an experience lasting more than one hour without any compensation. All these factors contributed to a very small set of tests, seven in total. Of these seven tests, two would not be considered valid for certain quantitative analyses (as will be explained in Section 4.4). However, they remain useful for qualitative observations and for metrics that are unaffected by the specific issues that compromised them. Given the small number and the limited quality of the data, we decided not to consider these data as statistically relevant. We therefore consider these tests an explorative investigation of the system, rather than a controlled experiment. In the analysis, we will therefore focus on the individual users’ experiences and on general information that can be derived from the collected data, with the awareness that most of these conclusions might change with larger and better datasets. We are nevertheless convinced that our analysis can serve as preliminary work for a more precise experimental setup. In Section 7, we will outline the ideal experimental setup, based on what we have learned from user engagement with the system. In Sections 5 and 6, we will show that our analysis produced at least one significant result that demonstrates the effectiveness of our architecture in at least one important aspect of the problem (namely, context load reduction).

## 4.2. Architecture bypass

To compare our solution with a simple LLM, we created a bypass of the architecture in order to be able to test an interaction with a simple LLM in the same context. The details of the prompt used for this bypass are available in Appendix B. Here we will outline the rationale behind the elements used to prompt the LLM, in order to give it all the information also available to the architecture. This of course does not include information that are a product of the architecture, such as the retrieved documents or, in the case of the extended architecture, the formatted input. The prompt for the generator under the architecture contains the following information:

- the natural language story outline;
- the previous generated scene;
- a summary of the scenes in this narrative Act;
- the current narrative act;
- the list of the actions that the user wants (their character) to perform;
- the information on the current room, derived from the formatted story outline;
- additional information from the retrieval system.

Most of this information would not be available in a common application that simply makes use of an LLM. Normally, such applications maintain the entire conversation with the user up to that point in their context. However, considering the setting of interactive narrative storytelling, it would make more sense to keep only the information regarding the generated story. Hence, we decided to include in the prompt for each scene all the previous generated scenes. We found that this is the only way to simultaneously ensure that no information is lost and to evaluate the unassisted LLM without any external help. In fact, in this way, the LLM without architecture will have more information about previous scenes than the LLM with architecture, since it will be able to access the entire story, rather than only a summary of previous events. The effects of this difference will be evident in the produced data.

Another important element is, of course, the story outline. As discussed in Section 3.4.2, this might be regarded as a critical point of the comparison. However, considering that the formatted outline is specifically tailored

to be used by the architecture, and that it could be generated by an extended version of the architecture, we consider it to be a specific advantage of the architecture. Hence, we provided to the LLM only the natural language story outline, in the prompt for each scene. Finally, in addition to the user input for each specific scene, we included the character’s description, goal, and motivation for every scene. These are, in fact, not products of the architecture, and it would have been unfair to omit them from the prompts of the simple LLM. However, a problem now arises. Our architecture is programmed to manage the pace of the story, adapting it to the user’s inputs. The architecture ensures (net of errors of the information extraction model) that the story moves on following the user’s intent, and that the story terminates only when the user has reached their goal. A normal LLM would instead continue the story indefinitely, even if the user has reached their goal, or, on the contrary, might lose track of the story and start looping. We need to understand how to handle this aspect of story generation with the LLM. The idea is that, in both cases, we want to impose a sense of progression on the story. In one case, this is done by the architecture. In the case of the simple LLM, we need to prompt it to do so. We therefore decided to proceed as follows: we impose a fixed number of scenes within which the story must terminate. For each scene, the prompt contains the current scene number over the total number of scenes. The LLM is instructed to manage the pace of the story so that it terminates exactly at the last scene. Finally, the system prompt of the generation model was modified to adapt it to work without the architecture. The modifications were made with the intent to change only the aspects strictly necessary to adapt it to the new task.

The application front end also went through a small revision. A disadvantage of not using the architecture was to not have the ability to understand when a Companion joined the character, or when the character obtains an Item. Hence, even if the story narrated that the character obtained a certain Item, this would have not been displayed by the application in the UI. To avoid that the user frustration for this fact could influence his judgment on the stories, we removed both the Companion and Inventory section of the Character Sheet (see Section 3.3.1).

### 4.3. Evaluation questionnaire

All users filled out a questionnaire after the end of the experience. The complete questionnaire is available in Appendix A. Here we provide an overview of the questionnaire, which will be useful in the following sections. The questionnaire is divided into four main parts:

- **User information.** The user is asked to provide some information regarding their own familiarity with RPGs, LLMs and Virtual Interactive Narrative Experiences (VINEs for short).
- **Short User Experience Questionnaire (UEQ).** This is a standard questionnaire, widely used in the literature to assess the quality of the interaction with the user. It consists of eight pairs of opposite adjectives referring to the application. The user must choose a value (among seven) between the two extremes [6].
- **System Usability Scale (SUS).** Another standard used in the literature for evaluating interactive applications. It asks the user to express their level of agreement with a set of statements on a scale from 1 to 5 (strongly disagree vs strongly agree) [59].
- **Story Evaluation (SE).** These are questions specifically designed for our application. They attempt to capture the user’s opinions on various aspects of the generated story, such as coherence, language and responsiveness to user input.

### 4.4. Conducted tests overview

In this section, we will discuss the user experiences, the immediate feedback from the users and their opinions afterward, and general qualitative considerations on the interactions and the quality of the generated stories. As anticipated, seven tests were conducted with a total of five users. We will refer to the users with capital letters to preserve their privacy: A, B, C, D, and E. All subjects were between 20 and 25 years old. Other information relevant to the tests was collected from the users and will be discussed for each of them when we analyze their experiences in detail. Two of these users, C and D, conducted two experiences each: the first employed the architecture, while the second used the bypass. A and B conducted only the test with the architecture. Their tests were interrupted prematurely, as they did not complete the experience in the intended way. Nevertheless, given the already limited amount of data, we will still use the results from these two experiences when possible, by analyzing the data up to the point of interruption and applying the appropriate methodological adjustments. In A’s case, an error in the architecture caused the story to loop back to the second act after its ending. B left the experience before its conclusion due to time constraints. Finally, E conducted the experience only with the bypass of the architecture. In the following, we will analyze the experiences of the users one by one, offering a qualitative view of the user interactions with the system that may guide future experimental setups and provide

some insight into the quality of the system.

#### 4.4.1 Test A

A was the first subject to test the application. His answers to the first part of the questionnaire are shown in Table 23. As anticipated in Section 4.4, this test is invalidated by an error that occurred in the application. When the user’s progression triggered the passage to the third act, a conditional branch error reloaded the rooms of the second act instead. The user, having a technological background, recognized that this was likely due to an application error rather than a flaw in the story itself, and decided to evaluate the story and the application without letting this detail affect their evaluation. Being the first interaction of an external user with the system, this test offered interesting insights into the system’s behavior. In this case, the user generally demonstrated the ability to engage with the system as intended. At the start of the test, he was instructed only to describe his character’s actions, not to write overly complex descriptions, and not to include an excessive number of actions in a single input. He deviated from these instructions only occasionally, and never in an excessive or disruptive manner. This aligns with his declared familiarity with RPGs, VINEs and LLMs.

Table 23: User ‘A’ – Familiarity and gaming frequency

| Question                                 | Answer    |
|------------------------------------------|-----------|
| RPG familiarity (1–5)                    | 3         |
| Generative AI (LLM) familiarity (1–5)    | 5         |
| Interactive narrative familiarity (1–5)  | 4         |
| Game frequency (RPG/interactive fiction) | Sometimes |

#### 4.4.2 Test B

This was the second test performed. B’s answers to the first part of the questionnaire are shown in Table 24. In this case, the familiarity of the user with RPGs and VINEs is less than the A’s, and in general this subject is the least familiar with interactive fantasy storytelling out of all the others. This is reflected in her approach to the application: in spite of having received the same instructions as the first user, she often deviated from the desired behavior. In particular, she often described or added additional elements, prompting the LLM with additional story elements that lies beyond the simple description of the character action. An example can be found on the fourth scene, the sixth scene of the story, where she inputted the following text:

*Dora was very curios about. So she decides to listen what Golgi has to said. he say something about a magic, the treasure was protected by a magic flower that keeps inside the key to open it. Dora doesn't know nothing about magic so ask also to Irina to take part of her adventure promising her a gem from the treasure.*

The "magic flower" mentioned by the user was not present in any way in the previous narrations, it’s completely invented by the user. Similar behaviors can be spotted throughout the experience, with the user sometimes directly inserting bits of story, instead of just describing their character’s action. This is likely not due to an intentionally harmful behavior of the user, rather with her inexperience with RPGs and interactive storytelling in general. The user was intentionally not corrected for these behavior, since our interest was to see how the system handled input not perfectly aligned with the intended one. In this case, these kinds of input significantly deviate from the intended use of the application, even if not in an excessive way. The system showed to be vulnerable to injection of unintended story elements, as expected. The architecture do not explicitly removes unintended story elements from the user input, and the LLMs do not distinguish well the differences between intended and unintended story elements. Moreover, a systematic removal of all the creative elements in the user inputs would ultimately hurt the interactive aspect of storytelling. To return to the present case, the LLM followed the story element of the "flower", and hallucinated a completely different course of events, that ultimately led the story to loop and stay locked on the first narrative act for the first fourteen scenes. After that, the story followed more or less the outline. However, the user decided to terminate the experience after scene eighteen, due to personal time constraints. The experience lasted in total 1:42 hours. This rendered this text not not valid from the point of view of statistical, parametric analysis. However, it was an interesting example of atypical interaction.

Table 24: User 'B' – Familiarity and gaming frequency

| Question                                 | Answer |
|------------------------------------------|--------|
| RPG familiarity (1–5)                    | 2      |
| Generative AI (LLM) familiarity (1–5)    | 5      |
| Interactive narrative familiarity (1–5)  | 2      |
| Game frequency (RPG/interactive fiction) | Rarely |

#### 4.4.3 Tests C1 and C2

This user, identified as 'C', conducted two experiments. The first test was with the architecture, while the second was without it. The two tests were conducted eight days apart. His answers to the first part of the questionnaire are shown in Table 25. The user described himself as only slightly more familiar with VINEs than B. However, he rarely produced inputs that significantly deviated from the intended format. He generally limited himself to describing his character's actions, or at most their intentions.

In the first experience, however, the model mistook an element of the character's backstory: in the character's description, the user had mentioned that the character had a group of followers, which was meant to be part of the character's motivation. However, the LLM mistakenly treated this group as active participants following the character. This issue had little to do with the architecture itself, and more to do with the instability of the underlying model. Nevertheless, we report this event because it may have influenced the user's judgment, especially since it did not occur in the second experience.

In the second experience (without the architecture), we set up the system to develop the story in the same number of scenes as the first experience (namely, 23). This choice was made both to reproduce a similar number of interactions with the user and to make the two stories comparable. The second experience was presented to the user as similar to the first. He was told that he would experience a story similar to the first, and that he might notice some changes, especially in the UI. In this way, the user was aware that the experience was different in some way from the first, but without any clue as to what the changes might be, except for the evident UI differences. Additionally, the user was instructed to use the same character (included goal and motivation) of the first experience.

In this second test, the LLM completely lost reference to the story outline. After the first few scenes, the LLM went off track. It did not mention three of the main events of the story (namely, Miriam the Warrior, Alken the Mad Knight and the Dragon of Ruins; see Appendix C). Moreover, it completely hallucinated entirely new events. Additionally, the story unfolded extremely rapidly, and the character had already obtained their goal and left the castle by scene 13. For the following 10 scenes, the user tried to finish the experience, and from his inputs entirely new events were hallucinated. This caused a fair amount of frustration and confusion for the user.

In general, the user was satisfied with both experiences. In the second one, he immediately recognized a greater degree of coherence in the story, but also a higher degree of repetitiveness, with entire sentences being reiterated almost verbatim throughout the entire story. This phenomenon could also be noticed in the first experience, but it was more prominent in the second and was immediately noticed by the user. However, the user expressed a general preference for the second experience, even though he acknowledged that the first one had a higher degree of creativity. Moreover, the user expressed appreciation for the functionalities for visualizing and handling Companions and Items in the first experience, underlining how the ability to visualize obtained Items and equip them during the unfolding of the story to increase specific characteristics was a major advantage of the first experience (and so, indirectly, of the architecture).

Table 25: User 'C' – Familiarity and gaming frequency

| Question                                 | Answer |
|------------------------------------------|--------|
| RPG familiarity (1–5)                    | 2      |
| Generative AI (LLM) familiarity (1–5)    | 4      |
| Interactive narrative familiarity (1–5)  | 3      |
| Game frequency (RPG/interactive fiction) | Rarely |

#### 4.4.4 Tests D1 and D2

The user 'D' also conducted two experiments, this time four days apart. His answers to the first part of the questionnaire are shown in Table 26. In this case, the user is very familiar with both RPGs and VINEs, but has low familiarity with LLMs. This was clearly visible from his inputs to the system: they were the most adherent to the ideal format. The user generally limited himself to describing his character's actions or thoughts, while introducing a higher degree of uniqueness and role-playing elements. He was also the one who challenged the system the most with absurd requests, and who added the most elements to his character during the course of the experience. However, this showcased the capabilities of the architecture to restrict the user's agency to a certain degree for the sake of the story—an essential aspect of immersive storytelling. The system was able to acknowledge the user's decisions and to describe, in a creative way, why certain actions would fail. This user was the most prolific in terms of interactions, producing a total of 30 scenes.

For the second experience, the user was briefed similarly to C: he was told that he would experience a story similar to the first, and that he might notice some changes, especially in the UI. Again, he was instructed to use the same character details as in the first experience.

Also in this case, the second experience followed the story outline less closely than the first one. The model hallucinated fewer new events, but it skipped or merged main events of the story. Additionally, also in this case the character reached his goal before the actual end of the experience, which made feel the user confused and incompleteness.

The user's impressions after the second experience resembled those of C to a striking degree. Like C, D expressed the view that the second story was more coherent, but also more repetitive, particularly towards the end. Particularly striking was the case of scenes 23, 24 and 25, which were literally identical (100% lexical overlap). However, like C, D generally preferred the second experience. Similarly, he shared the same opinion regarding the possibility of visualizing and manipulating Items.

Table 26: User 'D' – Familiarity and gaming frequency

| Question                                 | Answer |
|------------------------------------------|--------|
| RPG familiarity (1–5)                    | 5      |
| Generative AI (LLM) familiarity (1–5)    | 2      |
| Interactive narrative familiarity (1–5)  | 5      |
| Game frequency (RPG/interactive fiction) | Often  |

#### 4.4.5 Test E

E was the last tester of the application. He tested the system without the architecture directly. His answers to the first part of the questionnaire are shown in Table 27. He immediately noticed the repetitiveness of the generated stories. To accommodate the user's need for an experience that was not excessively long, we selected a total number of scenes equal to 20, which is still an acceptable number and close to the average number of scenes generated with the architecture so far (22.5 scenes). In this case, in contrast to the other two stories generated by the LLM alone, the story ended well before the character could even approach their goal. Also in this case, the LLM deviated from the original story outline. In addition to skipping main plot events, the LLM generated redundant and content-wise empty scenes, which slowed down the pace of the story. As in C's case, E often found himself wondering "what am I supposed to say now?" in response to inconclusive scenes that left little room for interaction.

Table 27: User E – Familiarity and gaming frequency

| Question                                 | Answer    |
|------------------------------------------|-----------|
| RPG familiarity (1–5)                    | 5         |
| Generative AI (LLM) familiarity (1–5)    | 3         |
| Interactive narrative familiarity (1–5)  | 4         |
| Game frequency (RPG/interactive fiction) | Sometimes |

#### 4.4.6 Wrap up and general comments on the tests

In general, the users showed appreciation and interest in the application and the idea behind it. One thing common to almost all users was that they tried to conjecture the mechanisms behind the generated story, even when no such mechanism actually existed (namely, in the no-architecture case). For instance, C believed he had missed a specific trigger in the second experience, as an explanation for why the story did not end with his character reaching their goal. Similarly, E correlated the obsessive reiteration of phrases regarding the "forest whispering secrets" with a trigger he had missed in the forest, prompting him to return there and heavily influencing the pacing of the story.

Overall, users seemed to prefer the experience without the architecture. However, this impression comes exclusively from C and D, both of whom tested the no-architecture system as their second experience. This may have introduced some bias, so a larger testing sample would be needed—and more importantly, tests with the order of experiences inverted—to truly assess whether the presence of the architecture influences the perceived quality of the experience.

What can be stated with greater certainty is that the architecture appears to have a negative impact on the perceived coherence of the stories, but also a positive impact on perceived diversity (meaning, less repetitiveness). This aligns with the intuition derived from the different prompting strategies of the two solutions: the architecture includes only the last scene and a summary of the previous scenes, thus losing some information about earlier story beats; on the other hand, the no-architecture solution includes all previous scenes in its prompt for each generation, providing the generator with the full context for every scene. However, this also causes the generator to constantly draw on previous scenes to produce new ones, entering a positive feedback loop that pushes it to fixate on common phrases and repeat them obsessively. This phenomenon accentuated with the increasing of the scenes, but was visible from the first few scene. Additionally, the pure LLM system showed a higher tendency to exceed the maximum number of characters allowed per scene and to deviate completely from the story outline.

In general, users appreciated the additional functionalities provided by the architecture (namely, the possibility to visualize and manipulate Items to change the characteristics of the character). The summary of the user responses to the first (personal) questions of the questionnaire is showed in Table 28.

Table 28: User familiarity and gaming frequency

| User ID | RPG Familiarity | LLM Familiarity | Narrative Familiarity | Game Frequency |
|---------|-----------------|-----------------|-----------------------|----------------|
| A       | 3               | 5               | 4                     | Sometimes      |
| B       | 2               | 5               | 2                     | Rarely         |
| C       | 2               | 4               | 3                     | Rarely         |
| D       | 5               | 2               | 5                     | Often          |
| E       | 5               | 3               | 4                     | Sometimes      |
| Mean    | 3.43            | 3.57            | 3.71                  | —              |

## 5. Quantitative Results of the Testing

In this section, we will analyze the quantitative results of the conducted tests, both from the users' questionnaires and from textual analysis of the produced text. For this section, it is worth reminding the reader of the limitations of the test campaign expressed in Section 4.1. Moreover, we did not use human-based or LLM-based metrics in this evaluation, again due to computational and time limitations. These limitations will be mentioned again whenever necessary, but we ask the reader to keep them in mind throughout the reading of this section.

### 5.1. Quantitative analysis of the user testing results

In this section, we will analyze the results of the questionnaires filled out by the users at the end of their experiences. Users C and D filled out the questionnaire for both of their experiences. Regarding the data on user familiarity with RPGs, VINEs and LLMs, we considered the ones from the first questionnaire, treating any changes in later questionnaires as repetition errors rather than actual indicators of improved familiarity. The remaining data were not modified in any way.

In Tables 29, 30, and 31, we summarize the results of the three questionnaire sections introduced in Section 4.3 and extensively shown in Appendix A, where the full text of the questions can be found associated with their respective labels. For brevity, the tables in this section show only the labels for each question. Each table reports

the mean, variance, standard deviation, minimum, and maximum values of each answer across all user tests, along with the overall results for each category and a corresponding normalized score to facilitate comparison. These results indicate an overall appreciation of the application and the stories, regardless of the presence of the architecture. By consulting Appendix A, the reader will notice that some questions in the SUS section (specifically, SUS2, SUS4, SUS6, SUS8, and SUS10) and one question in the SE section (SE14) have a negative connotation, meaning that a higher score actually corresponds to a lower degree of appreciation of the system. For these questions, the scores were therefore inverted: the score used in the tables was obtained by subtracting the original questionnaire score from 6 ( $\text{used\_score} = 6 - \text{question\_score}$ ). This ensured consistency across all questions.

Table 29: Descriptive statistics for UEQ (scale 1–7)

| Item                    | Mean         | Var.         | Std          | Min          | Max          | N        |
|-------------------------|--------------|--------------|--------------|--------------|--------------|----------|
| UEQ1                    | 5.429        | 2.619        | 1.618        | 2            | 7            | 7        |
| UEQ2                    | 6.571        | 0.286        | 0.535        | 6            | 7            | 7        |
| UEQ3                    | 5.429        | 1.619        | 1.272        | 3            | 7            | 7        |
| UEQ4                    | 5.571        | 3.619        | 1.902        | 3            | 7            | 7        |
| UEQ5                    | 5.429        | 0.619        | 0.787        | 5            | 7            | 7        |
| UEQ6                    | 6.714        | 0.238        | 0.488        | 6            | 7            | 7        |
| UEQ7                    | 6.571        | 0.619        | 0.787        | 5            | 7            | 7        |
| UEQ8                    | 5.714        | 0.571        | 0.756        | 5            | 7            | 7        |
| <b>OVERALL</b>          | <b>5.929</b> | <b>0.395</b> | <b>0.629</b> | <b>5</b>     | <b>7</b>     | <b>7</b> |
| <b>NORMALIZED (0-1)</b> | <b>0.821</b> | <b>0.011</b> | <b>0.105</b> | <b>0.708</b> | <b>1.000</b> | <b>7</b> |

Table 30: Descriptive statistics for SUS (scale 1–5) (scores reversed for negative items)

| Item                    | Mean         | Var.         | Std          | Min          | Max          | N        |
|-------------------------|--------------|--------------|--------------|--------------|--------------|----------|
| SUS1                    | 3.000        | 0.667        | 0.816        | 2            | 4            | 7        |
| SUS2                    | 5.000        | 0.000        | 0.000        | 5            | 5            | 7        |
| SUS3                    | 4.571        | 1.286        | 1.134        | 2            | 5            | 7        |
| SUS4                    | 5.000        | 0.000        | 0.000        | 5            | 5            | 7        |
| SUS5                    | 3.714        | 0.905        | 0.951        | 2            | 5            | 7        |
| SUS6                    | 4.000        | 2.000        | 1.414        | 1            | 5            | 7        |
| SUS7                    | 5.000        | 0.000        | 0.000        | 5            | 5            | 7        |
| SUS8                    | 4.571        | 0.619        | 0.787        | 3            | 5            | 7        |
| SUS9                    | 4.571        | 0.286        | 0.535        | 4            | 5            | 7        |
| SUS10                   | 1.000        | 0.000        | 0.000        | 1            | 1            | 7        |
| <b>OVERALL</b>          | <b>4.043</b> | <b>0.113</b> | <b>0.336</b> | <b>3</b>     | <b>4</b>     | <b>7</b> |
| <b>NORMALIZED (0-1)</b> | <b>0.761</b> | <b>0.007</b> | <b>0.084</b> | <b>0.600</b> | <b>0.875</b> | <b>7</b> |

Table 31: Descriptive statistics for Story Evaluation (scale 1–5) (scores reversed for negative items)

| Item                    | Mean         | Var.         | Std          | Min          | Max          | N        |
|-------------------------|--------------|--------------|--------------|--------------|--------------|----------|
| SE1                     | 4.000        | 0.333        | 0.577        | 3            | 5            | 7        |
| SE2                     | 3.714        | 1.238        | 1.113        | 2            | 5            | 7        |
| SE3                     | 4.571        | 0.619        | 0.787        | 3            | 5            | 7        |
| SE4                     | 4.000        | 1.333        | 1.155        | 2            | 5            | 7        |
| SE5                     | 4.429        | 0.286        | 0.535        | 4            | 5            | 7        |
| SE6                     | 4.000        | 1.333        | 1.155        | 2            | 5            | 7        |
| SE7                     | 4.714        | 0.238        | 0.488        | 4            | 5            | 7        |
| SE8                     | 4.000        | 0.333        | 0.577        | 3            | 5            | 7        |
| SE9                     | 4.000        | 1.333        | 1.155        | 2            | 5            | 7        |
| SE10                    | 4.857        | 0.143        | 0.378        | 4            | 5            | 7        |
| SE11                    | 4.000        | 0.667        | 0.816        | 3            | 5            | 7        |
| SE12                    | 3.714        | 0.905        | 0.951        | 2            | 5            | 7        |
| SE13                    | 3.143        | 0.476        | 0.690        | 2            | 4            | 7        |
| SE14                    | 3.429        | 0.952        | 0.976        | 2            | 5            | 7        |
| SE15                    | 3.571        | 1.619        | 1.272        | 2            | 5            | 7        |
| SE16                    | 3.143        | 0.143        | 0.378        | 3            | 4            | 7        |
| SE17                    | 4.000        | 0.000        | 0.000        | 4            | 4            | 7        |
| <b>OVERALL</b>          | <b>3.958</b> | <b>0.102</b> | <b>0.320</b> | <b>3.353</b> | <b>4.294</b> | <b>7</b> |
| <b>NORMALIZED (0-1)</b> | <b>0.740</b> | <b>0.006</b> | <b>0.080</b> | <b>0.588</b> | <b>0.824</b> | <b>7</b> |

We are, however, more interested in comparing the results between the tests that employed the architecture and those that did not. In Figures 8, 9 and 10 we show the results of the three sections of the questionnaire as histograms, grouped by architecture (Y vs N). In this case, we did not correct the results of the "negative" questions, but we distinguished them from the others by coloring their labels in red.

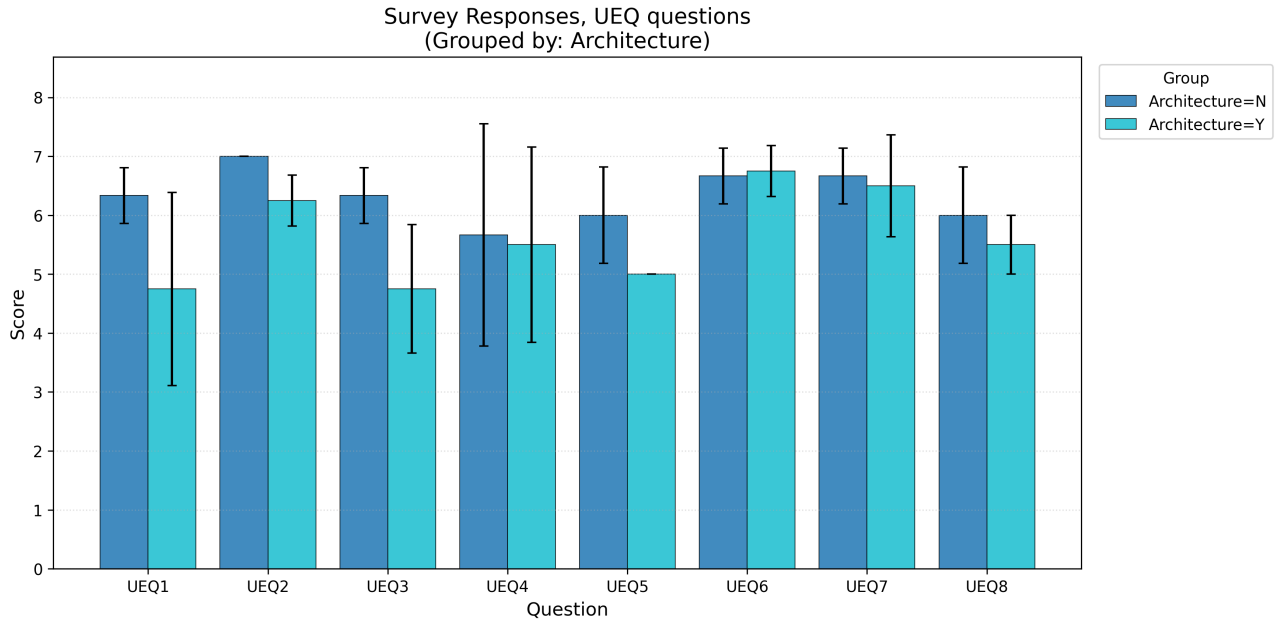


Figure 8: UEQ results

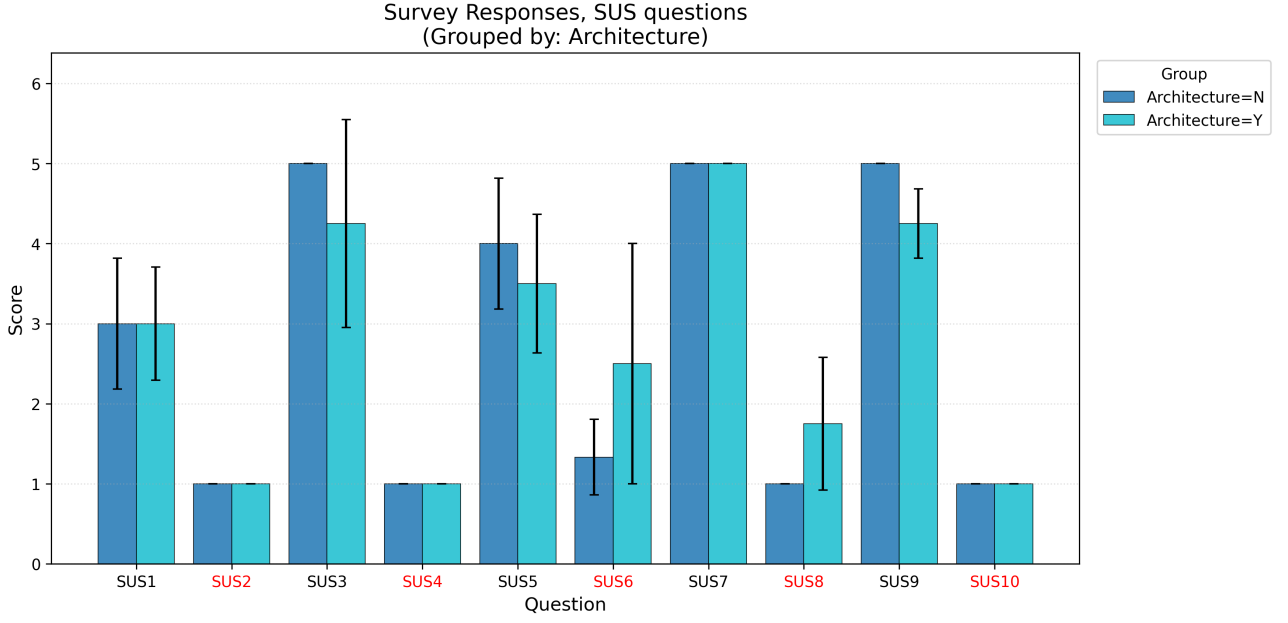


Figure 9: SUS results

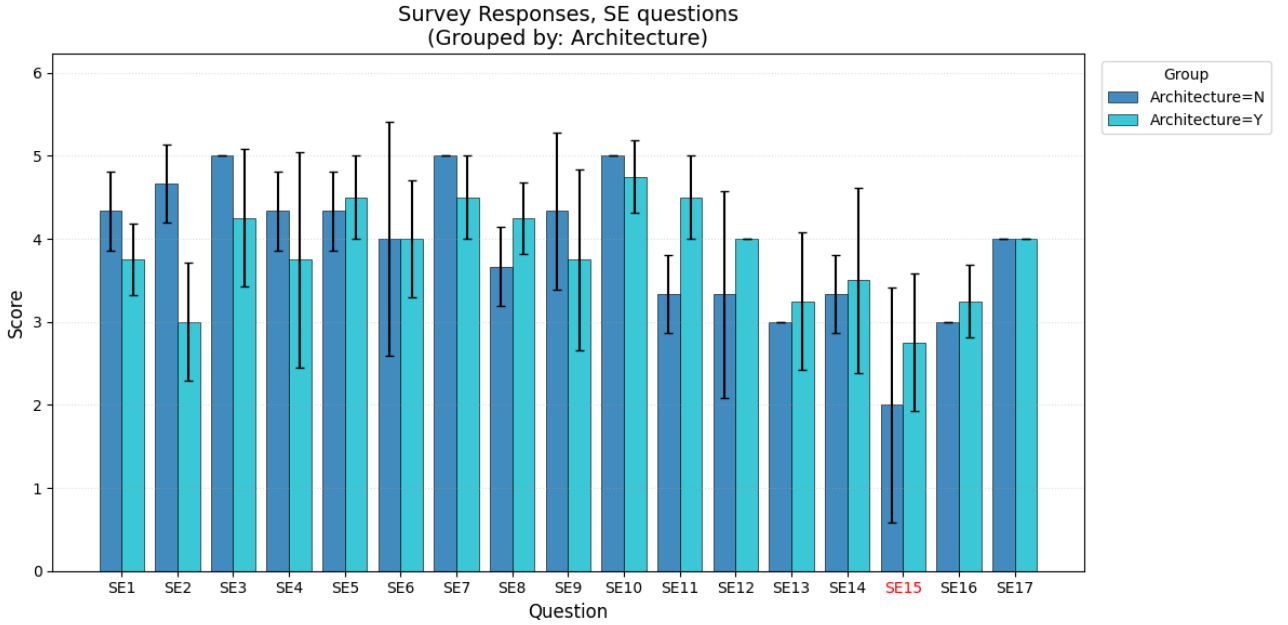


Figure 10: SE results

From the presented data, it seems that the users generally preferred the stories generated without architecture. However, to have a comprehensive view of the user opinions on the system with and without the architecture, we would like to have an easily interpretable score which measures the general quality of an experience. We deemed inappropriate the use of a simple mean among all the questions, since it would not take into account the different importance of each question and section. Hence, we designed a custom metric to evaluate the general quality of the users' experiences. The metric is designed in this way.

$$\text{FinalScore} = 0.25 \cdot \text{UEQ} + 0.25 \cdot \text{SUS} + 0.50 \cdot \text{SE}$$

Where UEQ, SUS and SE are the average normalized score for each part of the questionnaire. The SE score itself is a weighted average of three sub-components:

$$\text{SE} = 0.25 \cdot \overline{SE}_{\text{int}} + 0.25 \cdot \overline{SE}_{\text{glob}} + 0.25 \cdot \overline{SE}_{\text{other}}$$

where:

- $\overline{SE}_{\text{int}}$  is the mean of the interactivity-related questions (SE1, SE9, SE10, SE11, SE15);
- $\overline{SE}_{\text{glob}}$  is the sum of the global evaluation questions (SE16, SE17);
- $\overline{SE}_{\text{other}}$  is the mean of the remaining SE questions (SE2, SE3, SE4, SE5, SE6, SE7, SE8, SE12, SE13, SE14).

We will now explain the reasons behind these formulas. Let us begin with the formula of FinalScore. Here we weight the contribution of SE double with respect to the contributions of UEQ and SUS. This arise from the fact that we want to give a larger importance to the judgments about the narrative experience over those about the usability of the system. Regarding the weighing of the SE score, the rationale was the following:

- questions SE1, SE9, SE10, SE11 and SE15 measures the perceived interaction of the users. We want to give higher importance to these results;
- questions SE16 and SE17 asks global measurements of the perceived quality of the story and the experience, hence they need to be weighted more than a specific question on the quality of the story;
- questions SE2, SE3, SE4, SE5, SE6, SE7, SE8, SE12, SE13 and SE14 asks about specific aspects of the story (different from interactivity). They are weighted together and contribute to one fourth of the SE score.

The weight for this metric are arbitrarily choose according to the relative importance that we attributed to each individual score. The goal of this score is not to constitute a precise metric for statistical evaluations, but to offer an easily interpretable index of the perceived quality of the stories. The results of these computations for each story and aggregated by architecture are shown in Figure

Table 32: Individual scores per user.

| Test | Arch. | UEQ   | SUS   | SE    | Final | Interactivity | Overall Story | Overall Exp. | Other |
|------|-------|-------|-------|-------|-------|---------------|---------------|--------------|-------|
| A    | Y     | 0.708 | 0.600 | 0.706 | 0.680 | 0.850         | 0.500         | 0.750        | 0.722 |
| B    | Y     | 0.833 | 0.725 | 0.603 | 0.691 | 0.550         | 0.500         | 0.750        | 0.611 |
| C1   | Y     | 0.833 | 0.775 | 0.768 | 0.786 | 0.850         | 0.750         | 0.750        | 0.722 |
| C2   | N     | 0.896 | 0.800 | 0.751 | 0.800 | 0.950         | 0.500         | 0.750        | 0.806 |
| D1   | Y     | 0.708 | 0.775 | 0.751 | 0.747 | 0.950         | 0.500         | 0.750        | 0.806 |
| D2   | N     | 0.771 | 0.775 | 0.679 | 0.726 | 0.800         | 0.500         | 0.750        | 0.667 |
| E    | N     | 1.000 | 0.875 | 0.728 | 0.833 | 0.800         | 0.500         | 0.750        | 0.861 |

Table 33: Aggregated scores by architecture.

| Arch. | UEQ   |       |       | SUS   |       |       | SE    |       |       | Final |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|       | Mean  | Std   | Count | Mean  | Std   | Count | Mean  | Std   | Count | Mean  | Std   | Count |
| Y     | 0.771 | 0.072 | 4     | 0.719 | 0.083 | 4     | 0.707 | 0.074 | 4     | 0.726 | 0.050 | 4     |
| N     | 0.889 | 0.115 | 3     | 0.817 | 0.052 | 3     | 0.719 | 0.037 | 3     | 0.786 | 0.055 | 3     |

At first glance, these results seem to show a clear superiority of the no-architecture solution over the architecture. However, these data must be read through the lens of the limitations listed in Section 4.1. First, we can see that the larger differences between the two scores are given by the UEQ and SUS scores (0.118 and 0.098, respectively), while the difference between the SE scores is only 0.012. Inspecting the data for each experience leads to some interesting findings. First, for UEQ, we notice an outlier in the data corresponding to Test E. The score for this test is 1.000, meaning that the user gave the maximum score to all eight UEQ questions. Since this user tested only the no-architecture system, and the no-architecture experiences are only three, this outlier heavily influenced the mean score for that group. Also for SUS, Test E has the highest score in the sample. Interestingly, users C and D rated the usability of the no-architecture solution higher, despite both of them emphasizing that the extra functionalities offered by the first experience (the one with the architecture) were a very important part of the overall experience. This discrepancy might be due to the fact that they attributed great importance to the story (which they both considered better). However, this gap may be partly explained by a practice effect, a known bias whereby users tend to rate a system more favorably the second time they use it. Additionally, it is worth noting that another outlier appears in the Interactivity scores of Test B, which is considerably below the average of the other scores. This outlier belongs to the Architecture tests and substantially affects the SE score (considering that it accounts for one quarter of the score) This low score likely reflects B’s limited familiarity with interactive storytelling, as discussed in Section 4.4.2, rather than a flaw of

the architecture itself. Finally, we want to stress that the only score above 0.5 for the overall story comes from Test C1, which employed the architecture.

Considering these observations and the limitations expressed in Section 4.1, we wish to draw a conservative conclusion from these data:

*The data show a similar level of overall appreciation for the system, with a slight superiority of the no-architecture score, which might indicate a general preference of users for the solution without architecture. Nevertheless, the only score above 0.5 for overall story quality came from a test with the architecture (C1), and the SE scores were nearly identical between the two conditions. These observations suggest that the architecture may offer benefits that are not captured by the current dataset, and warrant further investigation with a larger sample. In conclusion, the amount of data collected is too small to allow strong general statements about the perceived quality of the stories.*

## 5.2. Quantitative analysis of the produced texts

In this section, we illustrate the results of the textual analysis performed on the produced narratives. As already stated, due to computational and time constraints, we did not use human-based or LLM-based metrics in this evaluation; rather, we based our analysis on lexical and semantic features. Purely lexical and semantic metrics (based on embeddings) can never capture the full complexity of a text’s quality, especially in narrative contexts, where quality depends heavily on human evaluation. However, our goal is not to provide a full evaluation of the generated text, but rather to gain some insight into the "objective" quality of the text and to compare these findings with the user evaluations. We evaluated the text using different metrics along four dimensions:

- **Diversity.** Measures the lexical and semantic variety of the text. There is no single optimal value for diversity: a value too low indicates that the text is repetitive and dull, while a value too high suggests that the text is too incoherent or hallucinated.
- **Adherence to story outline.** Measures how closely the generated story follows the story outline. In Section 3.4.2, we discussed how this aspect is important for the author of the story, but less so for the user experiencing it. Nevertheless, we decided to investigate this dimension to see how the architecture might affect it.
- **Adherence to user input.** Measures how well the output follows the user input. This is the most important dimension, since our application is specifically an interactive storytelling application, but also the most difficult to measure satisfactorily.
- **Coherence.** Measures the overall coherence of the story.

In the following sections, we will discuss each of these dimensions in depth, including the metrics used, how they were computed, our findings, and their limitations.

### 5.2.1 Diversity

Diversity refers to the lexical and semantic variety exhibited by a generated text. In the context of narrative generation, diversity is a crucial indicator of the richness and creativity of the story. A text with low diversity tends to be repetitive and dull, relying on a narrow set of words, phrases, or ideas throughout the narrative. This often results in a monotonous reading experience, where the user encounters the same expressions or sentence structures repeatedly. Conversely, a text with excessively high diversity may introduce too many unrelated concepts, leading to incoherence or hallucinated content that disrupts the narrative flow. Ideally, a well-balanced narrative should maintain a sufficient level of variety to keep the user engaged, while preserving enough consistency to ensure a coherent and believable story. We measured diversity using both a lexical and semantic approach.

As lexical measures we choose the Moving Average Type-Token Ratio (MATTR) for the general lexical diversity of the text. MATTR [19] is a modified version of the Type-Token Ratio metric, which measures the number of unique tokens over the total tokens of the text. Unlike the traditional Type-Token Ratio (TTR), which is known to be sensitive to text length, MATTR operates by computing the TTR over a sliding window of fixed size (typically 50 or 100 words) that moves through the text one word at a time. The final MATTR score is the average of these window-level TTR values. This approach makes MATTR robust to variations in text length, allowing reliable comparisons across stories of different lengths. MATTR is widely used in computational linguistics and text analysis, and has been implemented in several analysis frameworks [9, 17]. A value close to 1 indicates high lexical variety, while lower values reflect greater repetitiveness.

As a semantic measure, we employed the complement of the cosine similarity between embeddings (i.e.,  $1 - \text{similarity}$ ).

We applied these measurements at different levels of the generated stories:

- **Intra-scene diversity.** Measures the diversity within a single scene. Lexical diversity is computed using the vocabulary of each scene individually. For semantic diversity, the cosine similarity is computed

between the embeddings of each pair of sentences within the scene, and then averaged.

- **K-window diversity, with  $K = 2, 3, 4$ .** Measures the diversity across windows of  $K$  consecutive scenes. Lexical diversity is computed by concatenating the text of each tuple of  $K$  consecutive scenes. For semantic diversity, the cosine similarity is computed between the embeddings of each pair of entire scenes within the window.
- **Global diversity.** Measures the diversity across the entire story. Lexical diversity is computed using the vocabulary of the whole story (all scenes concatenated). For semantic diversity, the cosine similarity is computed between the embeddings of each pair of scenes, then averaged.

The plotting comparing architecture and no-architecture solutions for the in-scene and K-window lexical and semantic diversity are shown in Figure 11 and Figure 12. The histograms comparing (lexical and semantic) global diversity are instead shown in Figure 13a and Figure 13b. The Tables 34 shows the average scores of the global diversity, the average in-scene diversity and the average K-window diversity of each story. For these computations, we excluded the story A from the global computation (due to the occurred error in the architecture discussed in Section 4.4.1), while for the scene-by scene metrics, the story was truncated at scene 12, the last before the error occurrence.

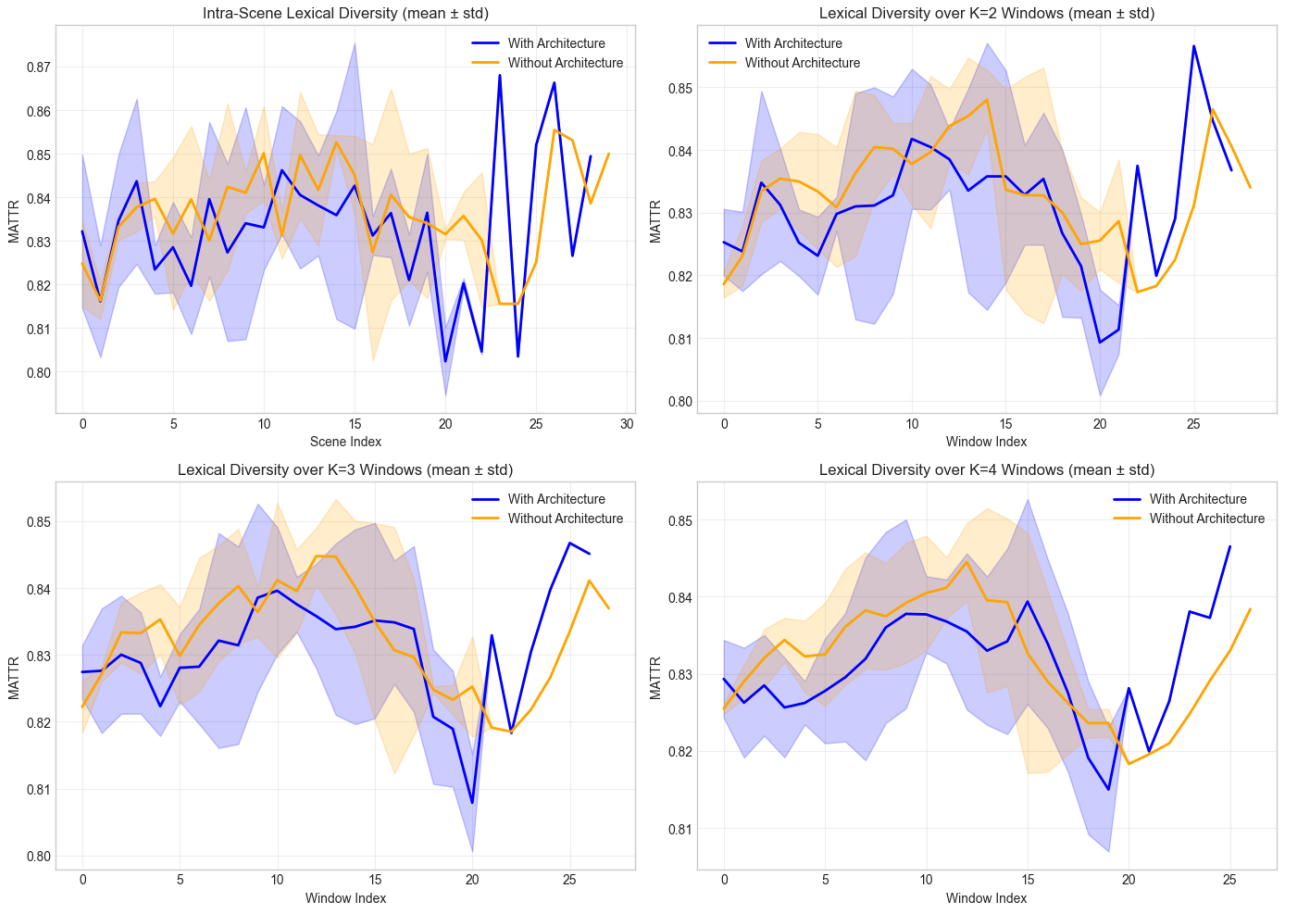


Figure 11: Lexical diversity plots for the stories.

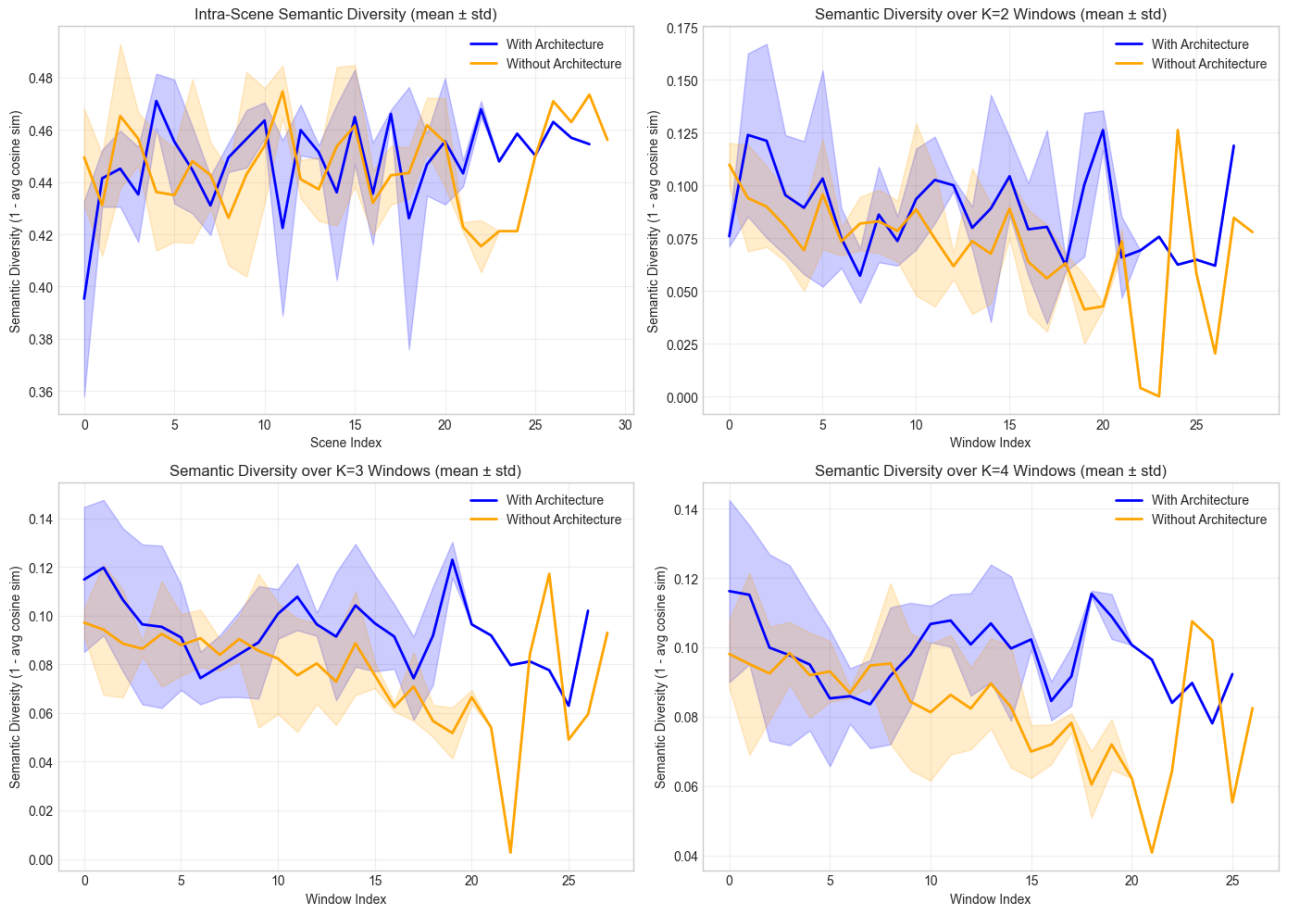
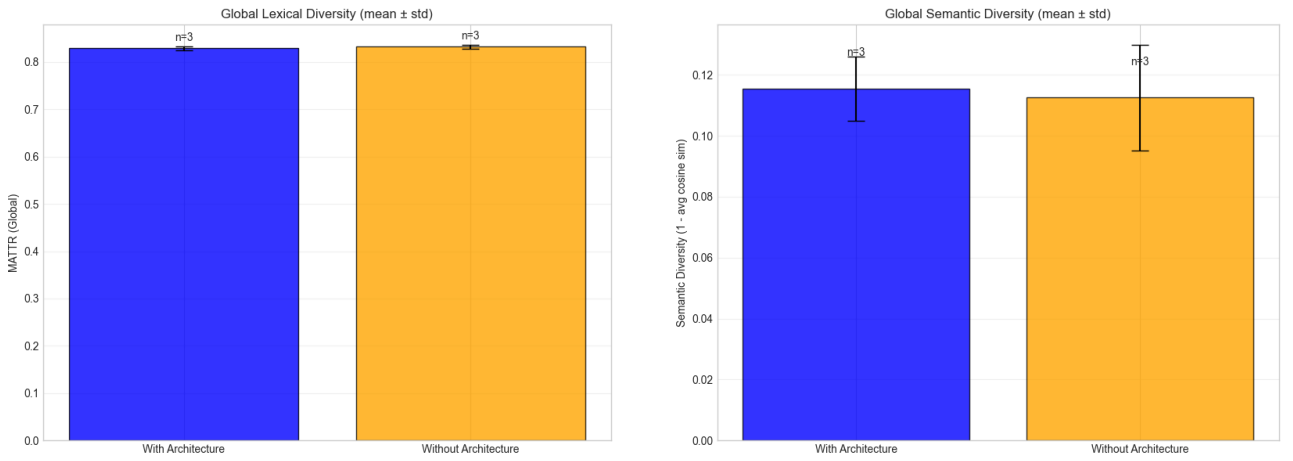


Figure 12: Semantic diversity plots for the stories.



(a) Lexical global diversity

(b) Semantic global diversity

Figure 13: Global diversity in comparison between architecture and no architecture.

Table 34: Composite scores for different alpha values and granularity levels (Formula:  $\alpha \cdot \text{Semantic} + (1 - \alpha) \cdot \text{Lexical}$ )

(a) Global

| Arch. | Alpha = 0.3 |        |        | Alpha = 0.5 |        |        | Alpha = 0.7 |        |        | N   |     |     |
|-------|-------------|--------|--------|-------------|--------|--------|-------------|--------|--------|-----|-----|-----|
|       | Mean        | Var    | Std    | Mean        | Var    | Std    | Mean        | Var    | Std    | 0.3 | 0.5 | 0.7 |
| No    | 0.6166      | 0.0001 | 0.0081 | 0.4726      | 0.0001 | 0.0116 | 0.3286      | 0.0002 | 0.0154 | 3.0 | 3.0 | 3.0 |
| Yes   | 0.6152      | 0.0000 | 0.0066 | 0.4724      | 0.0001 | 0.0081 | 0.3297      | 0.0001 | 0.0099 | 3.0 | 3.0 | 3.0 |

(b) Intra-scene

| Arch. | Alpha = 0.3 |        |        | Alpha = 0.5 |        |        | Alpha = 0.7 |        |        | N   |     |     |
|-------|-------------|--------|--------|-------------|--------|--------|-------------|--------|--------|-----|-----|-----|
|       | Mean        | Var    | Std    | Mean        | Var    | Std    | Mean        | Var    | Std    | 0.3 | 0.5 | 0.7 |
| No    | 0.7199      | 0.0000 | 0.0035 | 0.6415      | 0.0000 | 0.0051 | 0.5632      | 0.0001 | 0.0080 | 3.0 | 3.0 | 3.0 |
| Yes   | 0.7162      | 0.0000 | 0.0039 | 0.6392      | 0.0000 | 0.0034 | 0.5622      | 0.0000 | 0.0033 | 4.0 | 4.0 | 4.0 |

(c) K=2 windows

| Arch. | Alpha = 0.3 |        |        | Alpha = 0.5 |        |        | Alpha = 0.7 |        |        | N   |     |     |
|-------|-------------|--------|--------|-------------|--------|--------|-------------|--------|--------|-----|-----|-----|
|       | Mean        | Var    | Std    | Mean        | Var    | Std    | Mean        | Var    | Std    | 0.3 | 0.5 | 0.7 |
| No    | 0.6067      | 0.0000 | 0.0061 | 0.4547      | 0.0000 | 0.0065 | 0.3028      | 0.0000 | 0.0070 | 3.0 | 3.0 | 3.0 |
| Yes   | 0.6087      | 0.0000 | 0.0059 | 0.4608      | 0.0001 | 0.0077 | 0.3129      | 0.0001 | 0.0097 | 4.0 | 4.0 | 4.0 |

(d) K=3 windows

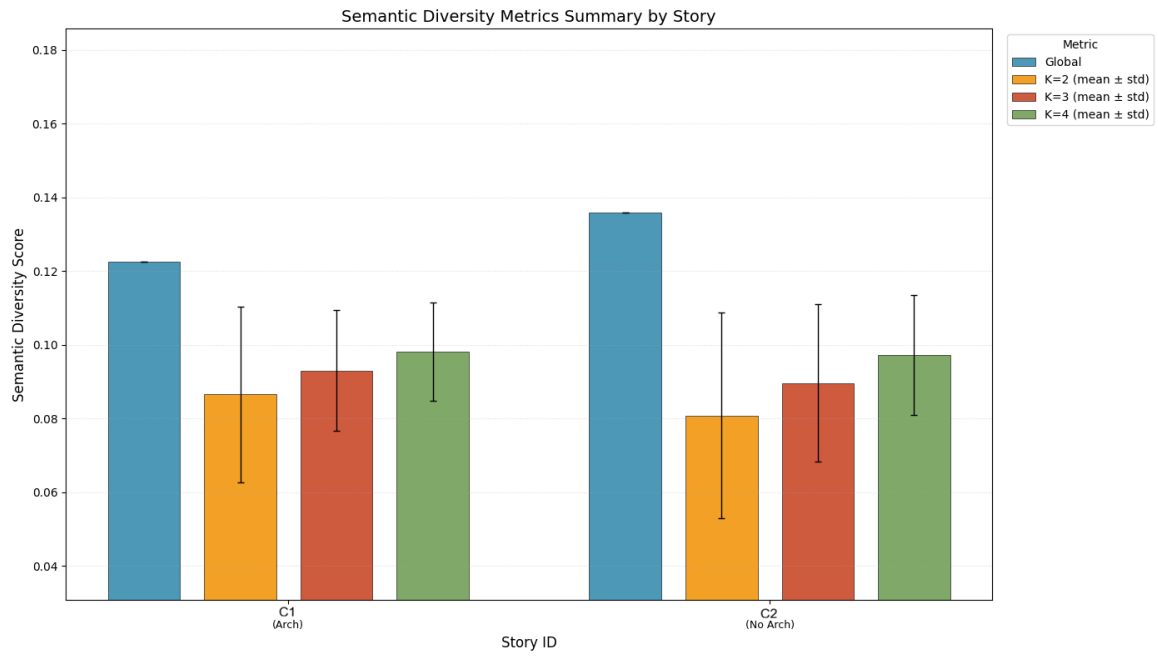
| Arch. | Alpha = 0.3 |        |        | Alpha = 0.5 |        |        | Alpha = 0.7 |        |        | N   |     |     |
|-------|-------------|--------|--------|-------------|--------|--------|-------------|--------|--------|-----|-----|-----|
|       | Mean        | Var    | Std    | Mean        | Var    | Std    | Mean        | Var    | Std    | 0.3 | 0.5 | 0.7 |
| No    | 0.6081      | 0.0000 | 0.0063 | 0.4573      | 0.0000 | 0.0070 | 0.3065      | 0.0001 | 0.0078 | 3.0 | 3.0 | 3.0 |
| Yes   | 0.6102      | 0.0000 | 0.0056 | 0.4635      | 0.0001 | 0.0072 | 0.3167      | 0.0001 | 0.0090 | 4.0 | 4.0 | 4.0 |

(e) K=4 windows

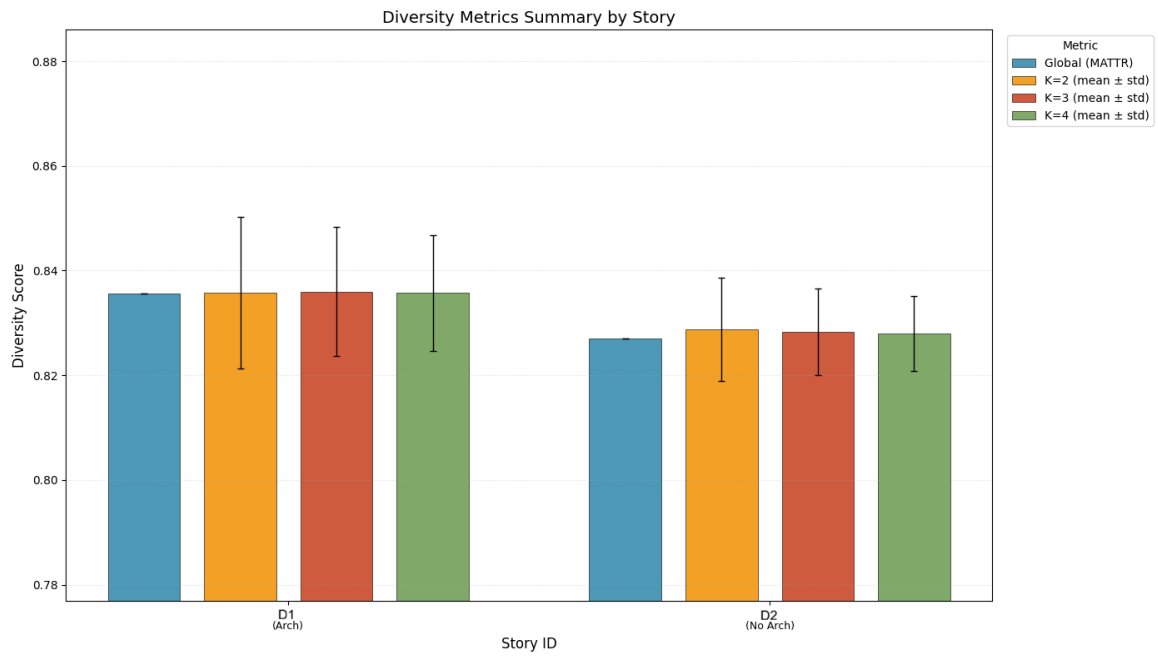
| Arch. | Alpha = 0.3 |        |        | Alpha = 0.5 |        |        | Alpha = 0.7 |        |        | N   |     |     |
|-------|-------------|--------|--------|-------------|--------|--------|-------------|--------|--------|-----|-----|-----|
|       | Mean        | Var    | Std    | Mean        | Var    | Std    | Mean        | Var    | Std    | 0.3 | 0.5 | 0.7 |
| No    | 0.6096      | 0.0000 | 0.0068 | 0.4598      | 0.0001 | 0.0077 | 0.3101      | 0.0001 | 0.0088 | 3.0 | 3.0 | 3.0 |
| Yes   | 0.6111      | 0.0000 | 0.0054 | 0.4649      | 0.0000 | 0.0069 | 0.3186      | 0.0001 | 0.0085 | 4.0 | 4.0 | 4.0 |

By analyzing the obtained scores, we did not find any relevant correlation between the use of the architecture and the measured diversity scores. A general observation is that we measure high lexical diversity and low semantic diversity, and that globally the architecture solution has slightly lower lexical diversity and slightly higher semantic diversity, but these differences are very small in magnitude ( $\sim 0.001$ ).

Comparing the two pairs of stories from the same users (C1 vs. C2, and D1 vs. D2), we observe that C2 has a higher global diversity than C1, but comparable K-window diversity, while D1 has greater diversity than D2 in both global and K-window diversity, albeit with a smaller magnitude than the differences between C2 and C1 for global diversity. These differences can be seen in Figure 14



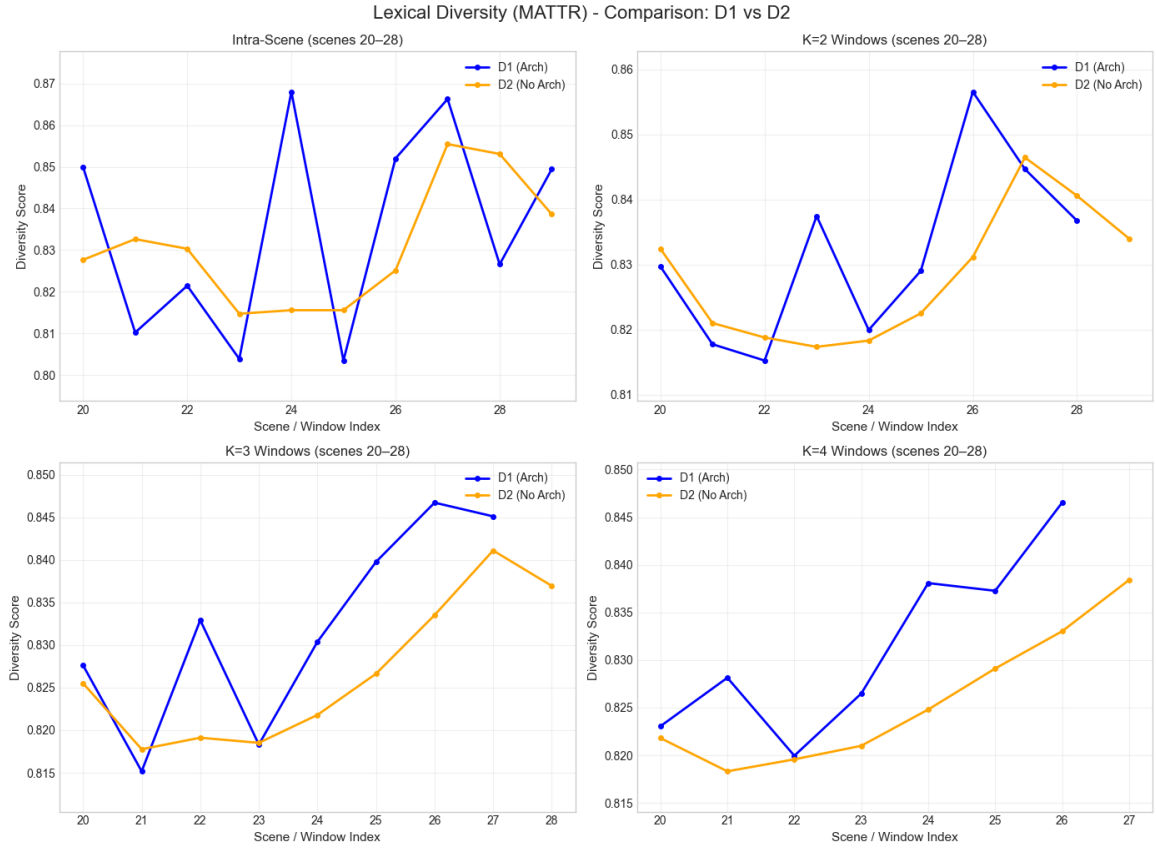
(a) C1 vs C2.



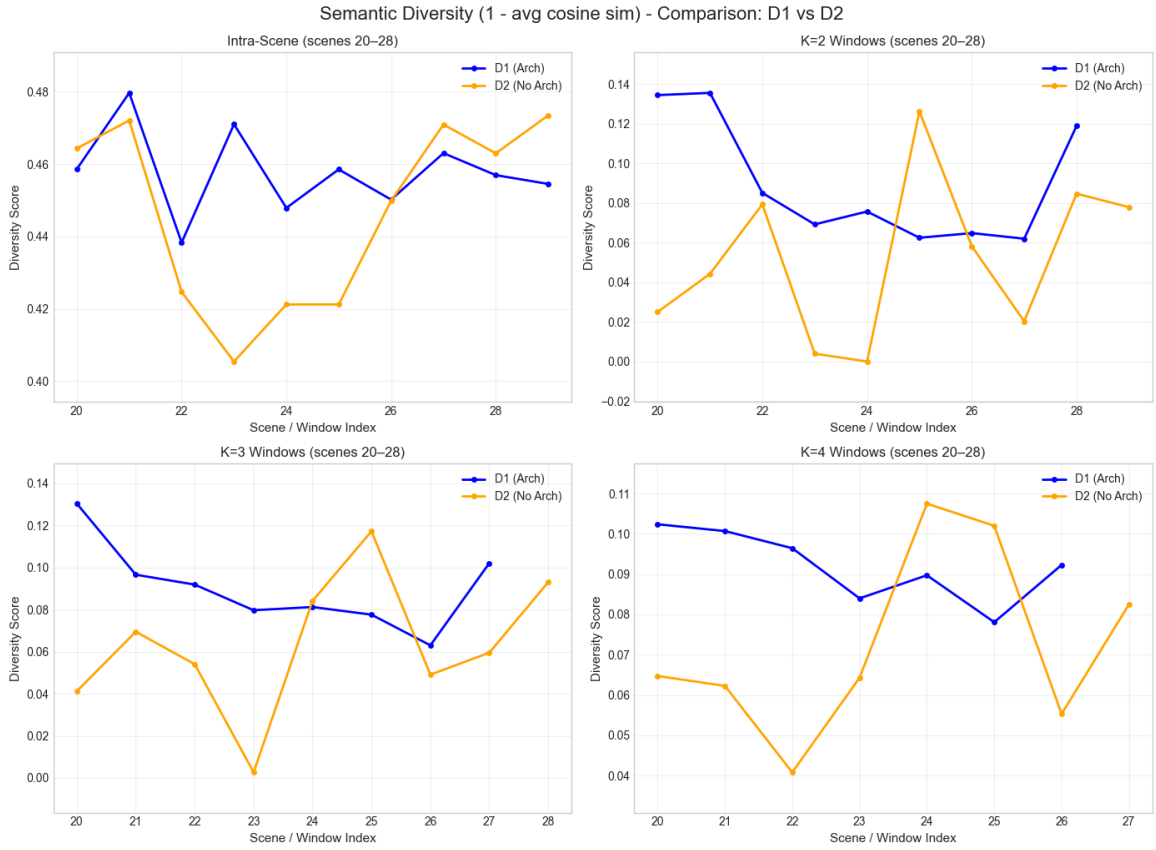
(b) D1 vs D2.

Figure 14: Diversity scores compared for users C and D

Moreover, after the 20th scene, D1 seems to begin surpassing D2 more consistently on all metrics, as shown in Figure 15.



(a) Lexical diversity past 20 scene (D1 vs D2)



(b) Semantic diversity past 20 scene (D1 vs D2)

Figure 15: Plotting of the diversity score of stories D1 and D2 past the 20th scene.

Especially for lexical diversity across scenes, D1 seems to outperform D2 more consistently. Moreover, for

the lexical diversity across 3-Windows and 4-Windows, D1 shows a clear pattern of dominance over D2. This might suggest—along with direct observation of the output from the D2 test—that after the 20th scene, the architecture tends to outperform the pure LLM in a more consistent manner, or rather that the pure LLM shows signs of degradation. However, with only one pair of stories substantially exceeding the 20-scene threshold, no statistically significant conclusion can be drawn from the diversity measurements.

### 5.2.2 Adherence to story outline

With the metrics of this dimension, we aim to measure how closely the generated story follows the provided story outline. To avoid that any accidental additional information in the formatted outline could influence the score, as well as to void the necessary data pre-processing that would be needed to make the formatted outline usable for lexical and semantic analysis, we used the natural language outline only for this metric. A simple similarity measure between the story and the outline would not suffice to capture the faithfulness of the story to the outline, since it would lose too much information and, more importantly, would not capture the temporal order of events. A story that includes all the elements of the outline, but in reverse order, would not be considered adherent to the outline. To address this problem, we computed adherence to the outline as follows:

- We split each story outline into sentences using `sent_tokenize` (NLTK).
- For each scene of a story, we compute the maximum cosine similarity between the embedding of the scene and each sentence of the story outline. This yields the outline sentence that best matches the scene, along with its similarity score.
- We compute the relative position of each outline sentence and each scene with respect to the center of the outline and the story, respectively, and then normalize these positions to the range  $[-1, 1]$ . Each scene and outline sentence is thus assigned a positional value between -1 and 1.
- We compute the difference between the positional value of each scene and that of its most similar outline sentence. This provides an index of the order discrepancy between the story outline and the generated text. For definition, the positional difference takes a value within the range  $[0, 2]$ .
- We complement these measurements with lexical metrics, specifically BLEU-1 (which measures overlap of single words), BLEU-2 (which measures overlap of bi-grams), and Coverage [33], which measures the average ROUGE-L recall between the scene and the corresponding outline sentence.
- We compute a comprehensive score as a weighted sum of the semantic, order, and lexical similarities between scenes and the outline.

This approach was inspired by the Order and Coverage metrics introduced in the paper *LOT: A Story-Centric Benchmark for Evaluating Chinese Long Text Understanding and Generation* [33], adapted here to the context of interactive storytelling in English. The key difference lies in the Order metric. The original Order from [33] checks for pairwise inversions between events in the generated story and those in the outline, providing a discrete measure of sequence correctness. In contrast, our adaptation computes the relative distance of each scene and each outline sentence from the center of their respective sequences, normalizing these positions to the range  $[-1, 1]$ . The discrepancy between the positions of matched scene–outline pairs yields a continuous measure of temporal misalignment, ranging from 0 (perfect alignment) to 2 (complete reversal). This approach captures not only inversions but also the degree of temporal shift, offering a more granular assessment of order adherence.

The plotting comparing architecture and no-architecture solutions for the lexical similarity, coverage, BLEU-1 and Positional difference in Figure 16, while the lot of the combine metric is shown in Figure 17. The Table 35 shows the average values of the single metrics and of the composite one.

The composite metrics is computed as a weighted sum between the single metrics, using the following formula:

$$0.35 \cdot \text{Semantic} + 0.25 \cdot \frac{(2.0 - \text{Positional Difference})}{2} + 0.2 \cdot \text{Coverage} + 0.1 \cdot \text{BLEU-1} + 0.1 \cdot \text{BLEU-2}$$

The weights are selected to reflect the order of importance that we attribute to each standalone metric.

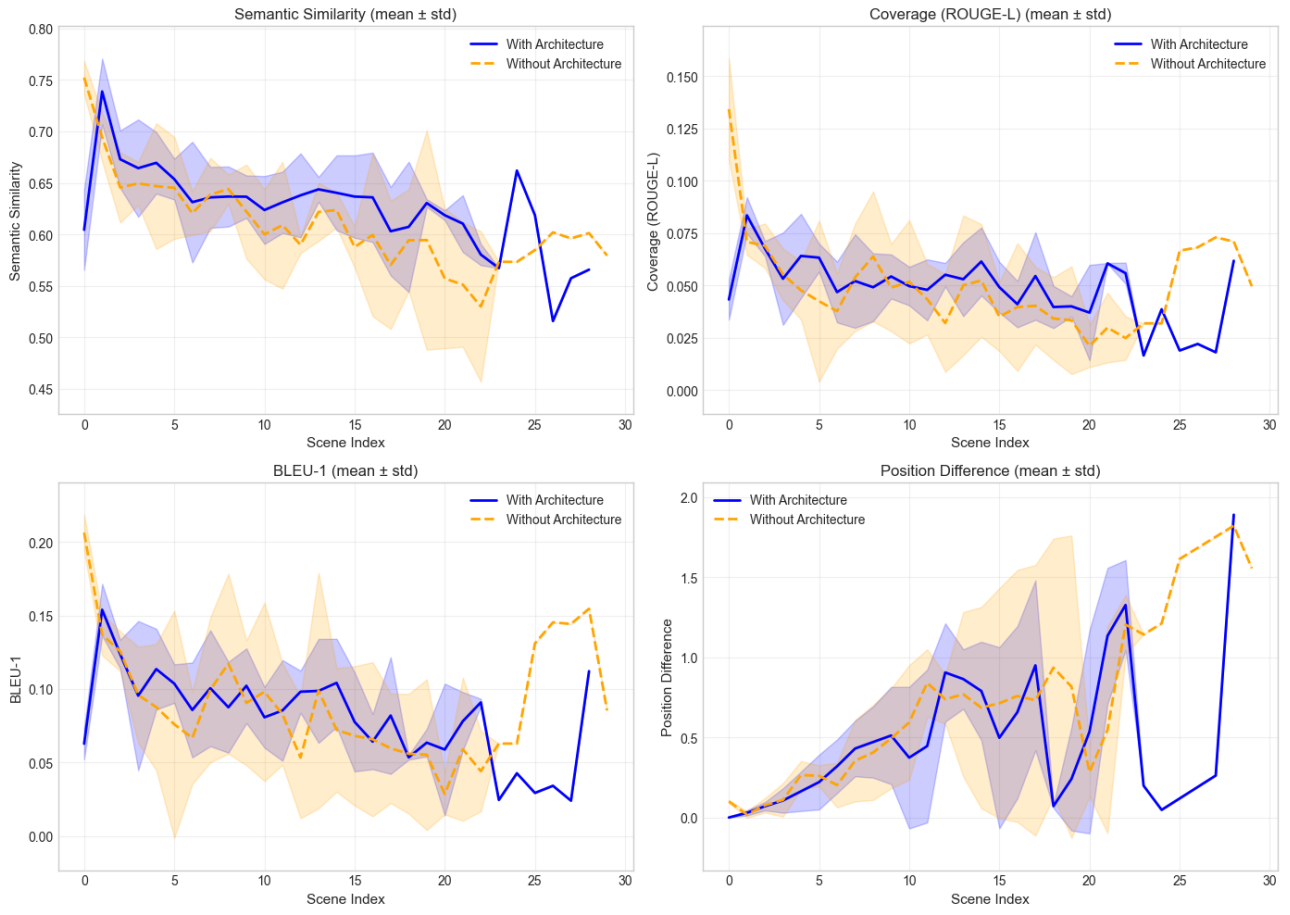


Figure 16: Story Outline Adherence scores for metric

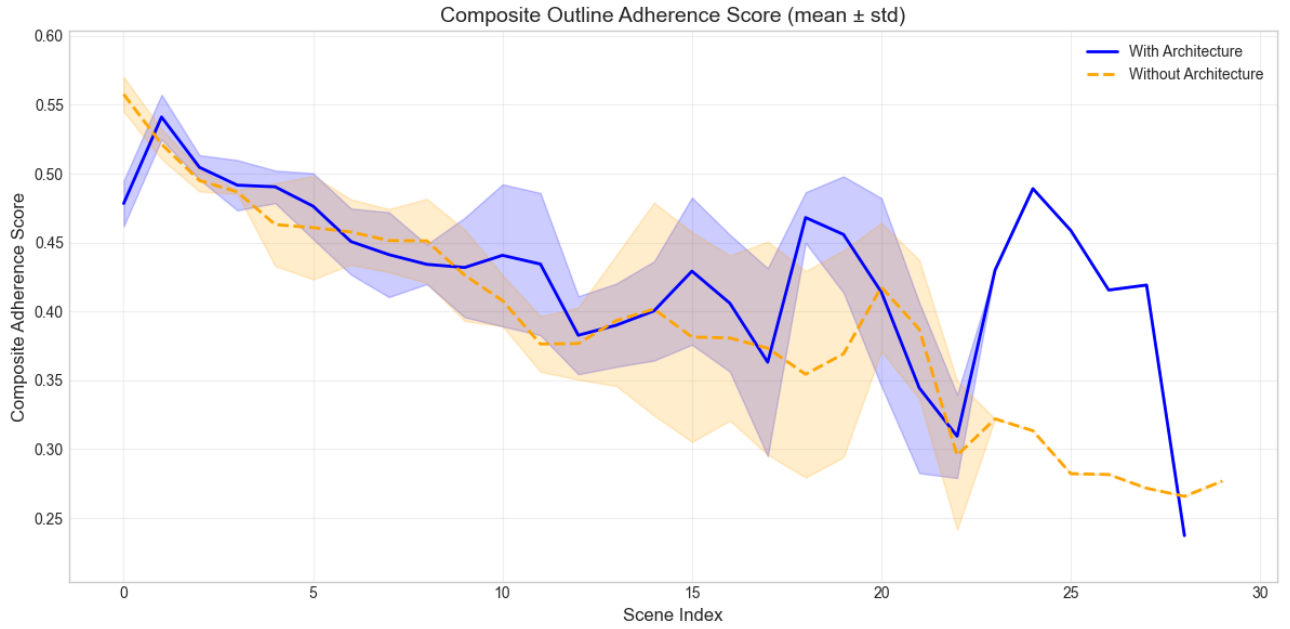


Figure 17: Combined Story Outline Adherence scores

Table 35: Outline adherence summary by story

| Metric              | Architecture |        |        |   | No Architecture |        |        |   | Diff    |
|---------------------|--------------|--------|--------|---|-----------------|--------|--------|---|---------|
|                     | Mean         | Var    | Std    | N | Mean            | Var    | Std    | N |         |
| Semantic Similarity | 0.6377       | 0.0005 | 0.0213 | 4 | 0.6193          | 0.0012 | 0.0353 | 3 | 0.0184  |
| Position Difference | 0.4193       | 0.0600 | 0.2450 | 4 | 0.6122          | 0.0918 | 0.3030 | 3 | -0.1929 |
| Coverage (ROUGE-L)  | 0.0522       | 0.0001 | 0.0082 | 4 | 0.0501          | 0.0001 | 0.0105 | 3 | 0.0020  |
| BLEU-1              | 0.0890       | 0.0002 | 0.0151 | 4 | 0.0889          | 0.0006 | 0.0240 | 3 | 0.0001  |
| BLEU-2              | 0.0182       | 0.0000 | 0.0029 | 4 | 0.0187          | 0.0000 | 0.0049 | 3 | -0.0005 |
| Composite Score     | 0.4420       | 0.0005 | 0.0219 | 4 | 0.4110          | 0.0005 | 0.0222 | 3 | 0.0309  |

Note: story A was truncated to 12 scenes, for the reasons discussed in Section 4.4.1. Mean, variance, and standard deviation are computed on per-story means.

In this case, even if no strong correlation can be drawn, it appears that the architecture slightly but consistently outperforms the no-architecture solution (except for BLEU-2, in which, however, it is outperformed only by a delta of 0.0005) on adherence to the story outline. The most notable improvement is the positional difference, which is reduced by the architecture by almost 0.2. Even though the improvement is modest, its persistence across various metrics and stories, along with its consistency with the direct observations discussed in Section 4.4, suggests that the architecture may improve the adherence of the generated story to the story outline.

### 5.2.3 Adherence to user input

This is probably the most important dimension for our system, but also one that is very hard to measure without using LLM-based techniques or human evaluation. We adopted the following procedure to compute a metric that could give us an indication of the quality of the stories in this dimension.

- First, we preprocessed the user’s input. Considering that the user input might contain actions narrated in the first person, we replaced first-person pronouns with the character’s name and adjusted the verbs accordingly. This was not performed on direct dialogues, which were left unchanged. Additionally, we split the user’s input into *action units* using strong punctuation as separators. Also in this case, direct dialogues were preserved as they were.
- For each *action unit*, we then computed the semantic and lexical similarity with each sentence of the corresponding scene. We combined these values for each sentence of the scene, and took the value with the maximum similarity for each *action unit*. The semantic similarity is computed as the cosine similarity between the embeddings of the sentences and the *action units*. The lexical similarity is the recall of the unit on the sentences, computed as the ratio between the intersection of sentence tokens and unit tokens, and the unit tokens.
- In addition to the total score, we defined two thresholds of activation for lexical and semantic similarity. If both thresholds are surpassed, then the action corresponding to the unit is considered activated by the scene.
- Finally, we computed the interaction score for each scene by averaging the scores of the individual action units, each weighted by the length of the tokens composing the action unit. The input is considered activated by the scene if at least one of the action units is activated (OR operator).

This metric is inspired by the CC3 metric from the paper *CML-BENCH: A Framework for Evaluating and Enhancing LLM-Powered Movie Scripts Generation* [87], adapted here to the context of interactive storytelling. In the original paper, CC3 is used to compare characters’ intentions—expressed through dialogue—with their subsequent actions, measuring how well the script aligns character motivation with behavior. We adopt a similar intuition, but shift the comparison to the user-system interaction level: instead of intentions and actions within the narrative, we compare the user’s input (action units) with the scene generated in response. This captures how faithfully the system follows the user’s directions, a dimension that is central to interactive storytelling but absent in static script generation.

The plotting of the average scores per scene is shown in Figure 20. Tables 36 show the average values for architecture vs no architecture, and for each story.

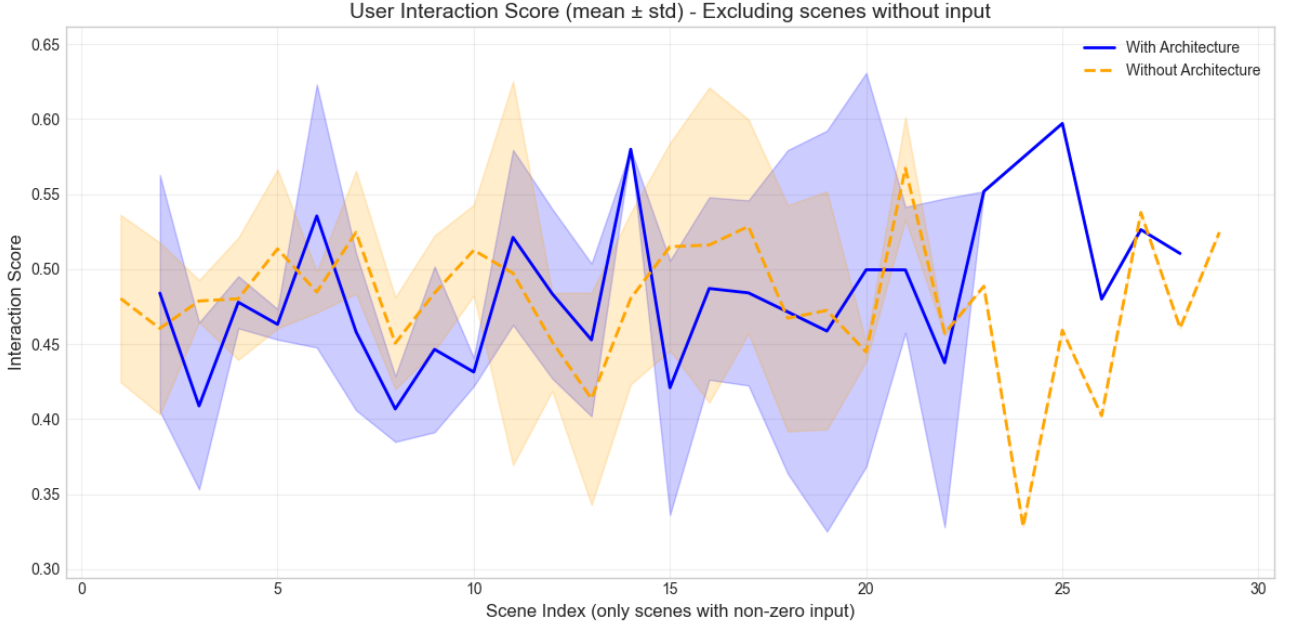


Figure 18: Plotted Average user interaction per scene across stories, comparing architecture vs no architecture

Table 36: User interaction metrics

(a) Aggregated by architecture

| Architecture         |                              | Interaction Score |        |        |             | Actuation %  |
|----------------------|------------------------------|-------------------|--------|--------|-------------|--------------|
|                      |                              | Mean              | Var    | Std    | N (stories) | Total scenes |
| With Architecture    | Mean per story               | 0.4714            | 0.0004 | 0.0211 | 3           | 57           |
|                      | Actuation % (mean per story) | 0.5047            | 0.0350 | 0.1871 | 3           | 57           |
| Without Architecture | Mean per story               | 0.4861            | 0.0012 | 0.0352 | 3           | 70           |
|                      | Actuation % (mean per story) | 0.5212            | 0.0079 | 0.0889 | 3           | 70           |

(b) Per story

| Story | Architecture         | Scenes with input | Mean Score | Actuation % |
|-------|----------------------|-------------------|------------|-------------|
| B     | With Architecture    | 13                | 0.4473     | 30.77       |
| C1    | With Architecture    | 19                | 0.4868     | 52.63       |
| C2    | Without Architecture | 22                | 0.5211     | 59.09       |
| D1    | With Architecture    | 25                | 0.4801     | 68.00       |
| D2    | Without Architecture | 29                | 0.4507     | 55.17       |
| E     | Without Architecture | 19                | 0.4864     | 42.11       |

Note: scenes with input score  $\leq 0.001$  were excluded as missing input. Story A was excluded entirely due to missing player inputs.

Once again, no strong correlation between the architecture and adherence to user input emerges from the data. On, average, the no-architecture solution seems to perform slightly better than the architecture. If we focus on the data from the two pairs of comparable stories (C1 vs. C2, and D1 vs. D2), we observe that in the first pair, the no-architecture solution dominates, while the opposite occurs in the second pair. The magnitude of the difference in means is similar for the mean score (respectively  $\sim 0.04$  and  $\sim 0.03$ ), but different on the actuation percentage (respectively  $\sim 0.07$  and  $\sim 0.13$ ). A substantial outlier can be identified in story B. This aligns with the user experience scores discussed in Section 5.1 and the user experience direct observation discussed in Section 4.4.2.

### 5.2.4 Coherence

This dimension measures the degree to which the produced story is logically coherent. Again, measuring coherence without using LLM-based or human evaluation cannot yield a deep and satisfactory assessment of this dimension, but only an approximate value that can guide some conservative considerations. We computed two coherence metrics at two different levels of the story: scene-level coherence and story-level coherence.

- For scene-level coherence, we adopted the approach of [87] for their PR1 metric. This metric computes coherence as the mean cosine similarity score between consecutive sentences.
- For story-level coherence, we observed through empirical testing with pairs of LLM-generated scenes that extending the PR1 approach to the embeddings of entire scenes was not satisfactory, as it lost fine-grained information essential for capturing coherence between scenes. We therefore adopted a more sophisticated method. We divided each scene into sentences. For each pair of consecutive scenes, we computed the full cosine similarity matrix between the embeddings of each sentence of the first scene and each sentence of the second scene. We then assigned to each sentence a positional weight, computed as the exponential function of the distance of the sentence from the "center" (i.e., the end of the first scene and the beginning of the second scene), multiplied by a fine-tunable parameter  $\alpha$ , which we set to 0.5. This gives more importance to sentences near the transition between scenes. The coherence metric is given by the average cosine similarity of each pair of scenes, weighted by the respective positional weights.

These metrics measure coherence as the semantic similarity between consecutive story units (sentences or scenes). However, this approach risks evaluating as highly coherent a story that exhibits a high degree of exact repetition of phrases. To avoid this, we added to the semantic similarity scores a penalty term that penalizes excessive lexical overlap between text units. The lexical overlap was computed using the Jaccard overlap, defined as the ratio between the intersection of the tokens and the union of the tokens of the two text units. The penalty function is a piecewise linear function of the Jaccard overlap, defined by two thresholds: the zero-threshold and the high-threshold. The function operates as follows:

- if  $\text{Jaccard} \leq \text{zero-threshold}$ , the penalty is always 0;
- if  $\text{zero-threshold} < \text{Jaccard} \leq \text{high-threshold}$  ( $= 0.6$ ), the penalty grows slowly with Jaccard;
- if  $\text{Jaccard} > \text{high-threshold}$ , the penalty grows rapidly with Jaccard.

We used  $\text{zero-threshold} = 0.2$  and  $\text{high-threshold} = 0.6$ . The penalty is then weighted by a penalty weight (which was set to 0.4) and subtracted from the similarity score to obtain the final coherence score. The overall score for the story is the weighted sum of the mean scene-level coherence score and the global story-level coherence score. This penalty represents an original addition to the PR1 metric from [87], which does not penalize excessive lexical repetitiveness.

The plotting of the average semantic similarity and lexical overlap both intra-scene and between consecutive scenes are showed in Figure 19, while the average scores per scene are shown in Figure 20. Tables 37 shows the average values of all the metrics and combined metrics per story, as well as for architecture vs no architecture.

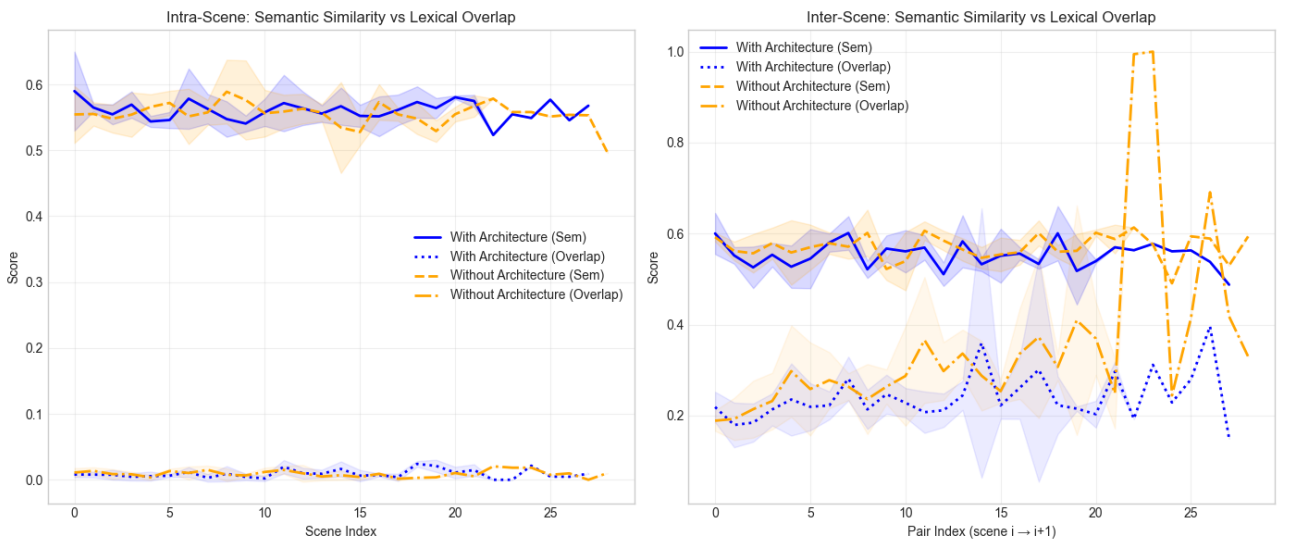


Figure 19: Plotted Average Semantic similarity and lexical overlap both intra-scene and between consecutive scenes, architecture vs no architecture.

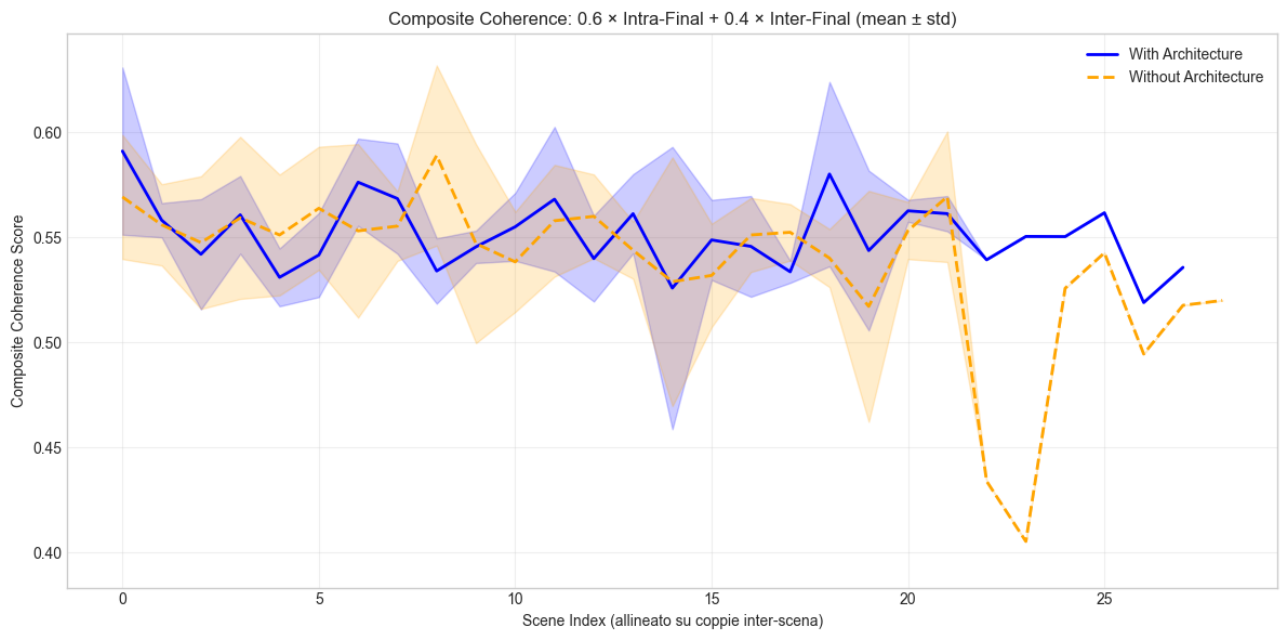


Figure 20: Plotted Average Combined Coherence scores per scene across stories, comparing architecture vs no architecture

Table 37: Coherence metrics

(a) Intra-scene metrics per story

| Story | Arch | Semantic |        |        | Overlap |        |        | Final  |        |        | Comb Score |
|-------|------|----------|--------|--------|---------|--------|--------|--------|--------|--------|------------|
|       |      | Mean     | Var    | Std    | Mean    | Var    | Std    | Mean   | Var    | Std    |            |
| A     | Yes  | 0.5541   | 0.0007 | 0.0256 | 0.0077  | 0.0000 | 0.0063 | 0.5541 | 0.0007 | 0.0256 | 0.5451     |
| B     | Yes  | 0.5529   | 0.0005 | 0.0222 | 0.0082  | 0.0001 | 0.0083 | 0.5529 | 0.0005 | 0.0222 | 0.5442     |
| C1    | Yes  | 0.5677   | 0.0009 | 0.0298 | 0.0074  | 0.0001 | 0.0077 | 0.5677 | 0.0009 | 0.0298 | 0.5612     |
| C2    | No   | 0.5754   | 0.0007 | 0.0256 | 0.0084  | 0.0000 | 0.0067 | 0.5754 | 0.0007 | 0.0256 | 0.5726     |
| D1    | Yes  | 0.5607   | 0.0006 | 0.0241 | 0.0104  | 0.0001 | 0.0089 | 0.5607 | 0.0006 | 0.0241 | 0.5528     |
| D2    | No   | 0.5502   | 0.0007 | 0.0270 | 0.0069  | 0.0000 | 0.0057 | 0.5502 | 0.0007 | 0.0270 | 0.5323     |
| E     | No   | 0.5420   | 0.0007 | 0.0261 | 0.0116  | 0.0000 | 0.0062 | 0.5420 | 0.0007 | 0.0261 | 0.5344     |

(b) Inter-scene metrics per story

| Story | Arch | Semantic |        |        | Overlap |        |        | Final  |        |        |
|-------|------|----------|--------|--------|---------|--------|--------|--------|--------|--------|
|       |      | Mean     | Var    | Std    | Mean    | Var    | Std    | Mean   | Var    | Std    |
| A     | Yes  | 0.5364   | 0.0026 | 0.0506 | 0.1992  | 0.0017 | 0.0412 | 0.5314 | 0.0022 | 0.0467 |
| B     | Yes  | 0.5586   | 0.0016 | 0.0403 | 0.2701  | 0.0211 | 0.1454 | 0.5312 | 0.0067 | 0.0816 |
| C1    | Yes  | 0.5608   | 0.0011 | 0.0325 | 0.2221  | 0.0022 | 0.0466 | 0.5516 | 0.0011 | 0.0329 |
| C2    | No   | 0.5867   | 0.0009 | 0.0303 | 0.2540  | 0.0070 | 0.0840 | 0.5686 | 0.0011 | 0.0336 |
| D1    | Yes  | 0.5575   | 0.0018 | 0.0423 | 0.2486  | 0.0051 | 0.0715 | 0.5410 | 0.0017 | 0.0417 |
| D2    | No   | 0.5679   | 0.0014 | 0.0368 | 0.3668  | 0.0429 | 0.2071 | 0.5056 | 0.0095 | 0.0975 |
| E     | No   | 0.5546   | 0.0014 | 0.0368 | 0.3013  | 0.0103 | 0.1016 | 0.5229 | 0.0010 | 0.0309 |

(c) Summary by architecture (Mean  $\pm$  Std across stories)

| Architecture | Metric                                                                           | Mean   | Std Dev | Variance |
|--------------|----------------------------------------------------------------------------------|--------|---------|----------|
| Yes          | Combined Coherence Score ( $0.6 \times \text{Intra} + 0.4 \times \text{Inter}$ ) | 0.5508 | 0.0080  | 0.0001   |
| No           | Combined Coherence Score ( $0.6 \times \text{Intra} + 0.4 \times \text{Inter}$ ) | 0.5465 | 0.0227  | 0.0005   |

Note: Combined Score is computed as  $0.6 \times \text{Intra\_Final} + 0.4 \times \text{Inter\_Final}$ . This arise from the idea that in-scene inconsistencies are more impactful on the perceived consistency of the story, especially in episodic narrations.

For coherence as well, we cannot find a very strong correlation with the use of the architecture. However, in this case, a slight but consistent pattern can be observed: the story generated without the architecture shows higher semantic similarity between scenes and sentences, but also a larger inter-scene lexical overlap. These differences, especially the semantic ones, are very small in magnitude and cancel out when the global penalized score is computed. Nevertheless, they align perfectly with the direct observations and the user’s impressions described in Section 4.4. These data suggest that the architecture seems to slightly reduce both the perceived coherence and the repetitiveness of the story.

### 5.3. Evaluation of the efficiency of the architecture

Until now, we have investigated the perceived and "objective" quality of the generated stories. However, there is another important aspect to evaluate: how does the architecture perform, in terms of consumed resources, with respect to the purely LLM-based approach? This is an important question to address, especially when working with small LLMs in low-resource environments, which is the case in our work. We explored two dimensions in which our architecture can make a meaningful difference in terms of performance: the waiting time between input and output, and the number of tokens in the context.

#### 5.3.1 Waiting and generation time

In this case, there is no sophisticated or complex metric to compute: we simply recorded, for each scene of each story, the waiting times between scenes. The plot of the mean total waiting time for scene can be seen in Figure 21a. The plot shows how the generation time of the scenes generated without the architecture becomes

increasingly larger than the entire processing time with the architecture. If we, however, imagine having access to a system with enough ram to run in parallel the Retrieval, Action Extraction and Path Taking path in parallel, then the real waiting time becomes the sum of the generation time and the max time between the three parallel operation. In this hypothetical scenario, realistically similar to the real use scenario, the comparison becomes even more striking, and is shown in Figure 21b. Tables 38 shows how, even on average, the waiting time of the architecture is less than that of the no-architecture case.

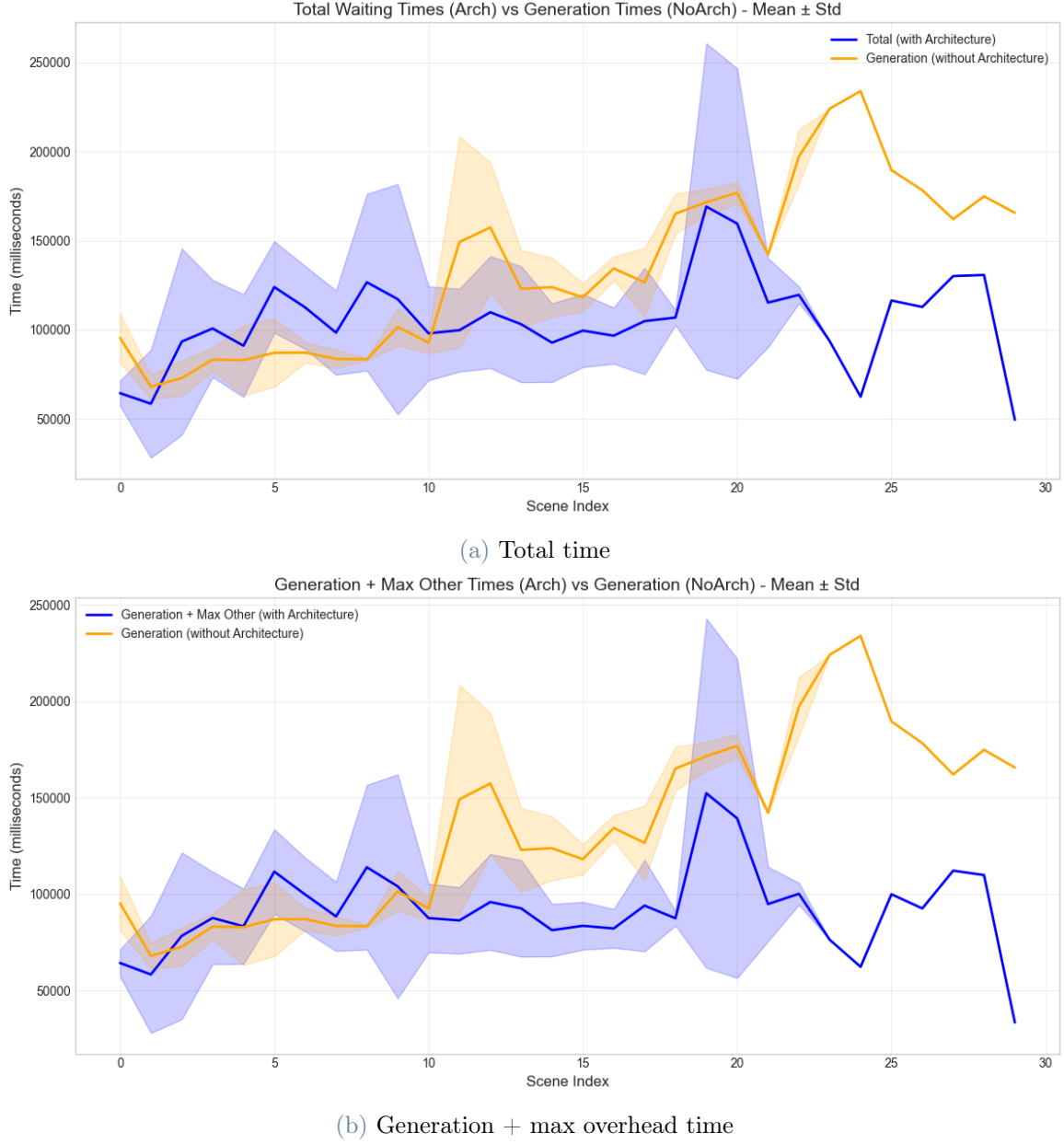


Figure 21: Comparison between processing time with and without the architecture.

Table 38: Timing summary

(a) Waiting times (mean over scenes)

| Metric                           | With Architecture | Without Architecture |
|----------------------------------|-------------------|----------------------|
| Total Waiting (mean over scenes) | 105243.33 ms      | –                    |
| Generation (mean over scenes)    | 91914.44 ms       | 135108.08 ms         |

Note: For the Without Architecture case, the total waiting time is not reported separately; it corresponds to the generation time.

### 5.3.2 Context tokens

For this measurement, we computed the total token count (system prompt + user prompt) provided as input to the LLM for the generation of each scene. The plot of the mean total token count of the context among all stories used to generate the scenes can be seen in Figure 22. These results are even more pronounced than those for waiting times: the context without the architecture grows linearly with the number of scenes, while it remains more or less constant with the architecture.

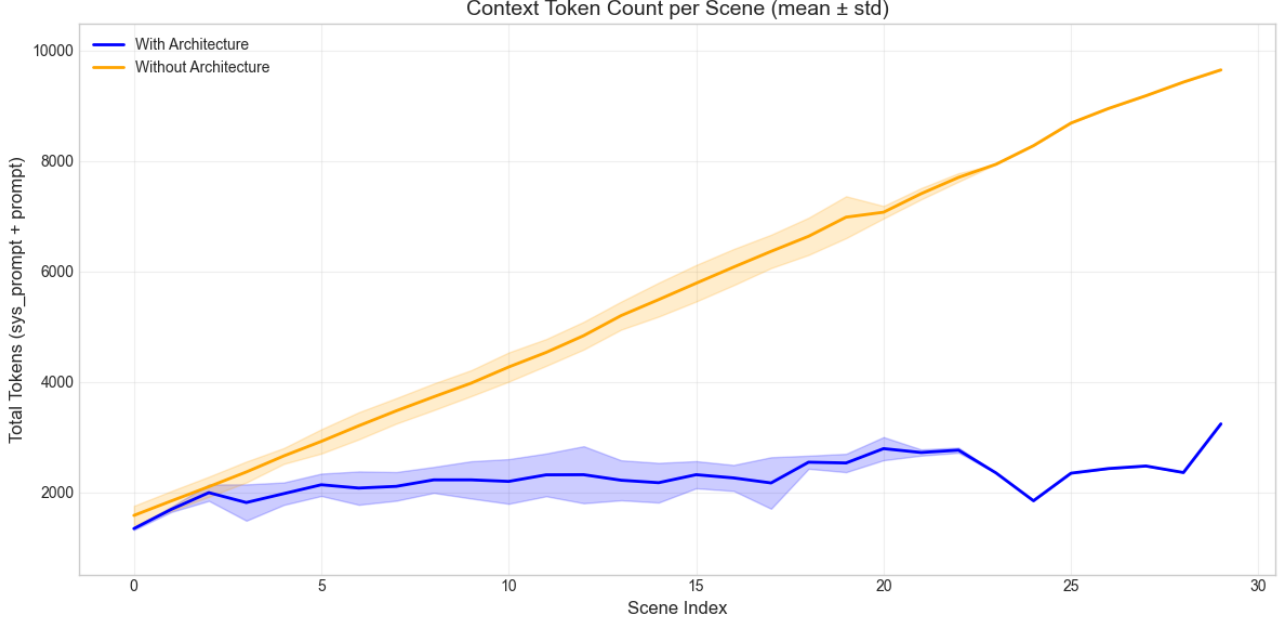


Figure 22: Comparison between token counts with and without the architecture.

By interpolating the growth of the context without architecture in function of the scenes, we obtain the function:

$$\text{context\_length}(\#scenes) = 282.7311 \cdot \#scenes + 1509.32$$

## 6. Conclusions

In this work, we set out to investigate whether a scaffolding architecture could improve the quality of interactive storytelling generated by small language models, particularly in terms of coherence, adherence to user input, and overall user experience. The results of our experiments allow us to draw several conclusions.

The constraints of our experimental setting, discussed in Section 4.1, prevented us from collecting enough data to obtain a statistically significant sample from which to draw strong conclusions. However, by combining direct observations of the users’ experiences, data from their personal evaluations, and objective metrics on the generated stories, we are nonetheless able to express some general observations, with the awareness that deeper and wider analyses might substantially alter these observations.

From the investigation and analysis conducted in Section 4 and Section 5, we argue that our findings suggest that the architecture does not appreciably degrade story quality, at least within the range of story lengths we were able to analyze. The main effects appear as a trade-off: a slight reduction in non-penalized semantic coherence, balanced by improvements in repetitiveness and adherence to the story outline. The observations from the tests and the user data support these claims, while also hinting at a possible slight decline of the perceived story quality. Overall, we support a conservative claim: the architecture maintains story quality while offering substantial performance advantages.

The data indicate that the architecture offers substantial performance advantages, keeping both processing time and context token length roughly constant, whereas in the system without the architecture, these two parameters grow linearly over time. These considerations, together with hints that the no-architecture system may start degrading after the 20th generated scene, prompt us to state the following:

*The advantage offered by the architecture is to allow a substantial reduction in processing time and context length, especially as the number of scenes in the story increases, without a substantial negative impact on story quality. The architecture therefore offers a considerable advantage for storytelling tasks in low-resource envi-*

ronments.

Other minor contributions of our work are:

- adaptation of PR1 and CC3 metrics from [87] and Order and Coverage metrics from [33] to our specific setting (interactive, scene-by-scene storytelling);
- explorative work on the testing of the application, which might help in constructing a more solid method of evaluation.

## 7. Limitations and Future Works

In this section, we discuss the main limitations of our work and outline possible directions for future research and development.

### 7.1. The author perspective

As discussed in Section 3.4.2, the current architecture relies on a manually crafted formatted story outline (JSON), rather than deriving it automatically from a natural language description. This introduces an additional human effort in the preparation phase, which may limit the practical applicability of the system, especially for authors who are not familiar with the structured format. While we argued that this is not a limitation of the architecture itself, but rather a constraint of the current implementation, it remains an aspect worth addressing in future work.

A natural extension would be the development of a dedicated component that converts a natural language story outline into the corresponding formatted JSON structure. This would require a sufficiently large language model, possibly fine-tuned for the task, and careful prompt engineering to ensure reliable extraction of narrative elements, events, and dependencies. Such a component would make the architecture more accessible to authors and facilitate a fairer comparison with purely LLM-based approaches, where the input is provided in natural language. Additionally, this would reduce the human bias introduced by manual formatting and allow for more systematic testing across a wider range of story outlines.

### 7.2. Limitations on the testing

As extensively discussed in Section 4.1, the experimental campaign suffered from several limitations that affect the generalizability of our findings. The most significant constraint was the small sample size, with only seven tests conducted across five users. This was mainly due to logistical difficulties: the application ran on a single local device, allowing only one test at a time, and the long duration of each session (between 1.5 and 2.5 hours) made it difficult to recruit volunteers, especially without financial compensation. Furthermore, the order of the experiences was not counterbalanced: users C and D tested the architecture first and the bypass second, potentially introducing a practice effect that may have inflated the scores of the no-architecture condition.

To obtain more robust and generalizable results, future testing should be conducted on a larger and more diverse sample. A key improvement would be the deployment of the application on a dedicated server, allowing multiple users to interact with the system simultaneously and in parallel. This would not only increase the number of tests that can be conducted within a given time frame, but also reduce the variability introduced by local hardware differences. Additionally, the experimental design should include counterbalancing: half of the users should experience the architecture first and the bypass second, while the other half should follow the reverse order. This would help isolate and control for order effects and practice-related biases, enabling a more accurate assessment of the architecture’s impact on story quality and user experience.

### 7.3. Limitations on the data analysis

As expressed in Section 5.2, we did not employ any human or LLM-based evaluation on the stories, mainly due to time and computational constraints. We instead employed purely lexical and semantic metrics. These metrics are easy and fast to compute, offer good insight into many aspects of the analyzed text, and have the advantage of being "objective", as they are based on pure mathematical computations. However, these methods fail to capture the complexities and nuances of a narrative text. Many aspects of a narrative text—such as logical coherence, character interest, emotional involvement, and story-wide themes—are either difficult or entirely impossible to capture through purely semantic or lexical means. To better evaluate the output of the presented application, we encourage the use of LLM-based or human-based metrics, to complement and integrate with lexical and semantic evaluations.

## 7.4. Parameter and hyperparameter tuning

Throughout this work, we adopted fixed values for the architectural parameters and the evaluation metrics thresholds, based on preliminary manual tuning and qualitative assessment. For instance, the window size for the MATTR metric, the coherence penalty thresholds, and the weighting coefficients of the composite quality scores were chosen heuristically rather than through a systematic optimization procedure. Similarly, the temperature and top-p parameters of the generation model were kept constant across all experiments, based on literature recommendations and limited pilot testing.

While these choices are justified by the exploratory nature of our study and the limited dataset available, they also represent a potential source of suboptimal performance. A more systematic tuning of both architectural and evaluation parameters could lead to more reliable and interpretable results, and potentially highlight configurations that better balance the trade-offs between efficiency and story quality.

Future work could explore automated or semi-automated approaches to parameter optimization, such as grid search or Bayesian optimization, applied to the complete pipeline. This would require a larger dataset and a more computationally efficient experimental setup, but could yield significant insights into the sensitivity of the architecture to its internal parameters and help identify optimal configurations for different use cases. Moreover, a careful tuning of the evaluation metrics themselves—for example, the penalty weights in the coherence metric or the thresholds for lexical overlap—could improve the alignment between automated scores and human judgments, making the evaluation framework more robust and informative.

## 7.5. Model fine-tuning

As clearly stated in Section 1, fine-tuning of the models lies outside the scope of this work. However, given the widely known benefits that careful fine-tuning procedures bring to LLMs, especially small ones, when they focus on a specific task, it is very likely that fine-tuning the models could significantly boost the capabilities of the architecture. While fine-tuning the generation model would likely be very difficult, since it would require careful data augmentation and quality control, training the models for information extraction would be much easier. The information extraction tasks are simple inference and deduction tasks, which produce straightforward results, sometimes as simple as a single number or a boolean. Generating synthetic data and evaluating model performance would be much easier, especially by means of powerful LLMs. This would allow even smaller models to be trained, making the architecture more performant both in terms of result quality and computational cost.

## 7.6. Used models

In this work, we relied solely on the use of the Llama-3.1 model from Meta, for the reasons expressed in Section 3.1. However, many other models, with different numbers of parameters or architectures, are available for free online, many of which might outperform Llama-3.1 on the specific tasks for which we employed it, even if they are ranked lower on benchmark leaderboards. This is particularly relevant for interactive storytelling, a task that is inherently difficult to evaluate through standardized benchmarks: metrics such as perplexity or accuracy on knowledge-based tests do not capture the subtleties of narrative coherence, immersion, and responsiveness to user input—qualities that depend heavily on human judgment and contextual adaptation. A model that excels on general-purpose benchmarks may still produce flat or repetitive narratives, while a smaller, less celebrated model could be more effective in this specific domain. We therefore encourage future research to experiment with different combinations of models for the various tasks of the architecture, and to explore evaluations that better reflect the qualitative dimensions of interactive storytelling.

# 8. Bibliography and citations

## References

- [1] Baldur's gate 3 — metacritic reviews & score. Metacritic — Official Game Page, 2023. Accessed: 2025-11-27.
- [2] Dungeons & dragons: Honour among thieves. Rotten Tomatoes — Official Film Page, 2023. Accessed: 2025-11-27.
- [3] The game awards 2023 — 2023 winners list. The Game Awards — Official Website, 2023. Accessed: 2025-11-27.

- [4] Marta Abba'. Dai videogame il boost tecnologico per un new normal davvero nuovo. *ZeroUno*, Aug 2022. Articolo online.
- [5] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-Time Rendering*. A K Peters/CRC Press, 4 edition, 2018.
- [6] Catherine Alberola, Götz Walter, and Henning Brau. Creation of a short version of the user experience questionnaire UEQ. *I-com*, 17(1):57–64, 2018.
- [7] Wenjuan Bu Anfeng Yang, Fei Kang. Tinysecgpt: Small-parameter llms can outperform large-parameter llms in cybersecurity. *Computers, Materials & Continua*, 87(2):–, 2026.
- [8] Gray Atherton, Rhys Hathaway, Ingela Visuri, and Liam Cross. A critical hit: Dungeons and dragons as a buff for autistic people. *Autism*, 29(2):382–394, 2025. PMID: 39166536.
- [9] Kenneth Benoit, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller, and Akitaka Matsuo. *quanteda: An R package for the quantitative analysis of textual data*, 2018.
- [10] Eden Bensaid, Mauro Martino, Benjamin Hoover, Jacob Andreas, and Hendrik Strobelt. Fairytaylor: A multimodal generative framework for storytelling. *ArXiv*, abs/2108.04324, 2021.
- [11] Aman Berhe, Camille Guinaudeau, and Claude Barras. Survey on narrative structure: from linguistic theories to automatic extraction approaches. *Traitement Automatique des Langues (TAL)*, 63(1):63–87, 2022.
- [12] Andrew Brown, Muhammad Roman, and Barry Devereux. A systematic literature review of retrieval-augmented generation: Techniques, metrics, and challenges. *Big Data and Cognitive Computing*, 9(12), 2025.
- [13] Sebastian Bruch, Siyu Gai, and Amir Ingber. An analysis of fusion functions for hybrid retrieval. *ACM Transactions on Information Systems*, 42(1):1–35, August 2023.
- [14] Jerome Bruner. *Acts of Meaning*. Harvard University Press, Cambridge, MA, 1990. ISBN 0674003608.
- [15] Martin Juan José Bucher and Marco Martini. Fine-tuned 'small' llms (still) significantly outperform zero-shot generative ai models in text classification, 2024.
- [16] Mingyue Cheng, Yucong Luo, Jie Ouyang, Qi Liu, Huijie Liu, Li Li, Shuo Yu, Bohou Zhang, Jiawei Cao, Jie Ma, Daoyu Wang, and Enhong Chen. A survey on knowledge-oriented retrieval-augmented generation, 2025.
- [17] Textrics Contributors. *Textrics: An R Package for Text Analysis*, 2024.
- [18] Eli Cook. Rearing children of the market in the “you” decade: Choose your own adventure books and the ascent of free choice in 1980s america. *Journal of American Studies*, 55(2):418–445, 2021. Open Access.
- [19] Michael A. Covington and Joe D. McFall. Cutting the gordian knot: The moving-average type-token ratio (MATTR). In *Pro ceedings of the 28th International Conference on Lexis and Grammar*, pages 153–162, 2008.
- [20] Andrea Yaoyun Cui and Pengfei Yu. Do language models have bayesian brains? distinguishing stochastic and deterministic decision patterns within large language models, 2025.
- [21] Tommoy Debnath, Md Nurul Absar Siddiky, Muhammad Enayetur Rahman, Prosenjit Das, and Antu Kumar Guha. A comprehensive survey of prompt engineering techniques in large language models. Technical report, TechRxiv, 2025.
- [22] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J.

- Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yudian Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025.
- [23] Vatsal Dhameliya. Annoy and efficient approximate nearest neighbor algorithm. <https://vatsal.website/blogs/annoy-and-efficient-approximate-nearest-neighbor-algorithm/>, 2023. Accessed: 2025-01-16.
- [24] Hervé Déjean, Stéphane Clinchant, and Thibault Formal. A thorough comparison of cross-encoders and llms for reranking splade, 2024.
- [25] Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrom, Roman Solomatin, Ömer Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Mariya Hendriksen, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Šuppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani, Pranjal Chitale, Simone Tedeschi, Nguyen Tai, Artem Snegirev, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri Krishnakumar, Anna Maksimova, Silvan Wehrli, Maria Tikhonova, Henil Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Hongjin Su, Jimmy Lin, Howard Yen, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muennighoff. Mmtb: Massive multilingual text embedding benchmark. *arXiv preprint arXiv:2502.13595*, 2025.
- [26] Evgenii Evstafev. The paradox of stochasticity: Limited creativity and computational decoupling in temperature-varied llm outputs of structured fictional data, 2025.
- [27] David M. Ewalt. *Of Dice and Men: The Story of Dungeons & Dragons and the People Who Play It*. Scribner, New York, NY, 2014.
- [28] Izzul Fatawi, Muhammad Roil Bilad, and Muhammad Asy’ari. The role of prompt engineering in enhancing llms: a systematic review of applications and ethical implications. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 15(2):1071–1086, 2026.
- [29] Robyn Fivush. The development of autobiographical memory. *Annual Review of Psychology*, 62:559–582, 2011.
- [30] Spandan Garg, Vikram Nitin, and Yufan Huang. Terminus-4b: Can a smaller model replace frontier llms at agentic execution tasks?, 2026.
- [31] Google. Understand llm sizes. <https://web.dev/articles/llm-sizes>, 5 2024. web.dev, Google for Developers. Accessed: 2026-07-02.
- [32] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur

Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yearly, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vitor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenxin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuwei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkan Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippas Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damla, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Laverder A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark,

Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanachandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Rutu Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024.

- [33] Jian Guan, Zhuoer Feng, Yamei Chen, Ruilin He, Xiaoxi Mao, Changjie Fan, and Minlie Huang. LOT: A story-centric benchmark for evaluating chinese long text understanding and generation. *Transactions of the Association for Computational Linguistics*, 10:434–451, 2022.
- [34] Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions, 2024.
- [35] Santiago Góngora. *Approaches to Interactive and Improvisational Storytelling*. PhD thesis, 08 2025.
- [36] Santiago Góngora, Luis Chiruzzo, Gonzalo Méndez, and Pablo Gervás. Payador: A minimalist approach to grounding language models on structured data for interactive storytelling and role-playing games. 06 2024.
- [37] Charlie Hall. How dungeons & dragons sparked a revolution in how we play just about everything, 2022. Accessed: 2025-11-27.
- [38] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *CoRR*, abs/1904.09751, 2019.
- [39] Parag Jain, Priyanka Agrawal, Abhijit Mishra, Mohak Sukhwani, Anirban Laha, and Karthik Sankaranarayanan. Story generation from sequence of independent short descriptions. *CoRR*, abs/1707.05501, 2017.
- [40] Ehsan Kamalloo, Nandan Thakur, Carlos Lassance, Xueguang Ma, Jheng-Hong Yang, and Jimmy Lin. Resources for brewing beer: Reproducible reference models and statistical analyses. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’24, page 1431–1440, New York, NY, USA, 2024. Association for Computing Machinery.
- [41] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.
- [42] Cameron Koch. *Baldur’s gate 3 continues its awards streak with 4 bafta wins, including best game*. *GameSpot*, 2024. Accessed: 2025-11-27.
- [43] Saar Kuzi, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. Leveraging semantic and lexical matching to improve the recall of document retrieval systems: A hybrid approach, 2020.
- [44] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 08 2019.

- [45] Label Your Data. Llm model size: 2026 comparison chart & performance guide. <https://labeleyourdata.com/articles/llm-fine-tuning/llm-model-size>, 12 2024. Accessed: 2026-07-02.
- [46] Lujun Li, Lama Sleem, Niccolo’ Gentile, Geoffrey Nichil, and Radu State. Exploring the impact of temperature on large language models:hot or cold?, 2025.
- [47] Erik Lindholm, John Nickolls, Stuart Oberman, and James Montrym. Nvidia tesla: A unified graphics and computing architecture. *IEEE Micro*, 28(2):39–55, 2008.
- [48] Microsoft, :, Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, Dong Chen, Dongdong Chen, Junkun Chen, Weizhu Chen, Yen-Chun Chen, Yi ling Chen, Qi Dai, Xiyang Dai, Ruchao Fan, Mei Gao, Min Gao, Amit Garg, Abhishek Goswami, Junheng Hao, Amr Hendy, Yuxuan Hu, Xin Jin, Mahmoud Khademi, Dongwoo Kim, Young Jin Kim, Gina Lee, Jinyu Li, Yunsheng Li, Chen Liang, Xihui Lin, Zeqi Lin, Mengchen Liu, Yang Liu, Gilsinia Lopez, Chong Luo, Piyush Madan, Vadim Mazalov, Arindam Mitra, Ali Mousavi, Anh Nguyen, Jing Pan, Daniel Perez-Becker, Jacob Platin, Thomas Portet, Kai Qiu, Bo Ren, Liliang Ren, Sambuddha Roy, Ning Shang, Yelong Shen, Saksham Singhal, Subhojit Som, Xia Song, Tetyana Sych, Praneetha Vaddamanu, Shuohang Wang, Yiming Wang, Zhenghao Wang, Haibin Wu, Haoran Xu, Weijian Xu, Yifan Yang, Ziyi Yang, Donghan Yu, Ishmam Zabir, Jianwen Zhang, Li Lyna Zhang, Yunan Zhang, and Xiren Zhou. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras, 2025.
- [49] Markus Montola. The invisible rules of role-playing: The social framework of role-playing process. *International Journal of Role-Playing*, 1(1):22–36, 2008.
- [50] MTEB Consortium. Massive text embedding benchmark (mteb) leaderboard. [http://mteb-leaderboard.hf.space/?benchmark\\_name=MTEB%28Multilingual%2C+v%29](http://mteb-leaderboard.hf.space/?benchmark_name=MTEB%28Multilingual%2C+v%29), 2024. Accessed: 2025-12-15.
- [51] Minh Nhat Nguyen, Andrew Baker, Clement Neo, Allen Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent llm outputs, 2025.
- [52] Eric Nichols, Leo Gao, and Randy Gomez. Collaborative storytelling with large-scale neural language models. In *Proceedings of the 13th ACM SIGGRAPH Conference on Motion, Interaction and Games*, MIG ’20, New York, NY, USA, 2020. Association for Computing Machinery.
- [53] Ageliki Nicolopoulou, Shayne Trapp, Bernadette Delgado, et al. Promoting oral language and narrative skills in preschool children through storytelling and story-acting. *Early Childhood Research Quarterly*, 31:147–162, 2015.
- [54] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert, 2020.
- [55] Strong National Museum of Play. The influence of dungeons and dragons on video games, 2021. Accessed: 2025-11-27.
- [56] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madeleine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin,

Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

- [57] John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.
- [58] Max Peeperkorn, Tom Kouwenhoven, Dan Brown, and Anna Jordanous. Is temperature the creativity parameter of large language models?, 2024.
- [59] Sebastian A. C. Perrig, Nick von Felten, Beat Vollenwyder, and Klaus Opwis. Development and psychometric validation of a positively worded German version of the system usability scale (SUS). *International Journal of Human-Computer Interaction*, 41(16):10399–10419, 2025.
- [60] Ronak Pradeep, Yuqi Liu, Xinyu Zhang, Yilin Li, Andrew Yates, and Jimmy Lin. Squeezing water from a stone: A bag of tricks for further improving cross-encoder effectiveness for reranking. In Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty, editors, *Advances in Information Retrieval*, pages 655–670, Cham, 2022. Springer International Publishing.
- [61] PwC. Global entertainment & media outlook 2025–2029, 2025. Press release, July 24, 2025.
- [62] Maila D. H. Rahiem. Storytelling in early childhood education: Time to go digital. *International Journal of Child Care and Education Policy*, 15(1):4, 2021.
- [63] Chandra Ramamurthy, Peixin Zuo, Gregory Armstrong, and Karl Andriessen. The impact of storytelling on building resilience in children: A systematic review. *Journal of Psychiatric and Mental Health Nursing*, 31(4):525–542, 2024.
- [64] Farouq Sammour, Jia Xu, Xi Wang, Mo Hu, and Zhenyu Zhang. Responsible ai in construction safety: Systematic evaluation of large language models and prompt engineering, 2024.
- [65] Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. In *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 155–161, 2024.
- [66] Gal Shachaf, Alon Brutzkus, and Amir Globerson. A theoretical analysis of fine-tuning with linear teachers, 2021.
- [67] Nisha Ingrid Simon and Christian Muise. Tattletale: Storytelling with planning and large language models.
- [68] Shriyansh Singh and Pramit Saha. Meta learning for efficient fine tuning of large language models. *International Journal of Scientific and Research Publications (IJSRP)*, 14(07):15106, 2024.
- [69] Spotify AB. Annoy: Approximate nearest neighbors oh yeah. <https://github.com/spotify/annoy>, 2024. Accessed: 2025-01-16.

- [70] Jan Stryszewski, Wiktor Prosowicz, Tomasz Kawiak, and Adrian Jaśkowiec. Aghir at longeval: Improving scientific information retrieval with dense representations and cross-encoder re-ranking. In *Proceedings of CLEF 2025: Conference and Labs of the Evaluation Forum*, volume 4038 of *CEUR-WS.org*, pages 3495–3503, Madrid, Spain, 2025. CEUR Workshop Proceedings. Accessed: 2025-12-16.
- [71] Richard Stubbs and Nikolas Sorensen. Tabletop role-playing games and social and emotional learning in school settings. *Social and Emotional Learning: Research, Practice, and Policy*, 5:100090, 2025.
- [72] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul R. Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Jack Krawczyk, Cosmo Du, Ed Chi, Heng-Tze Cheng, Eric Ni, Purvi Shah, Patrick Kane, Betty Chan, Manaal Faruqi, Aliaksei Severyn, Hanzhao Lin, YaGuang Li, Yong Cheng, Abe Ittycheriah, Mahdis Mahdih, Mia Chen, Pei Sun, Dustin Tran, Sumit Bagri, Balaji Lakshminarayanan, Jeremiah Liu, Andras Orban, Fabian Gra, Hao Zhou, Xinying Song, Aurelien Boffy, Harish Ganapathy, Steven Zheng, HyunJeong Choe, goston Weisz, Tao Zhu, Yifeng Lu, Siddharth Gopal, Jarrod Kahn, Maciej Kula, Jeff Pitman, Rushin Shah, Emanuel Taropa, Majd Al Merey, Martin Baeuml, Zhifeng Chen, Laurent El Shafey, Yujing Zhang, Olcan Sercinoglu, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anas White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, Alexandre Frechette, Charlotte Smith, Laura Culp, Lev Proleev, Yi Luan, Xi Chen, James Lottes, Nathan Schucher, Federico Lebron, Alban Rustemi, Natalie Clay, Phil Crone, Tomas Kocisky, Jeffrey Zhao, Bartek Perz, Dian Yu, Heidi Howard, Adam Bloniarz, Jack W. Rae, Han Lu, Laurent Sifre, Marcello Maggioni, Fred Alcober, Dan Garrette, Megan Barnes, Shantanu Thakoor, Jacob Austin, Gabriel Barth-Maron, William Wong, Rishabh Joshi, Rahma Chaabouni, Deeni Fatiha, Arun Ahuja, Gaurav Singh Tomar, Evan Senter, Martin Chadwick, Ilya Kornakov, Nithya Attaluri, Iaki Iturrate, Ruibo Liu, Yunxuan Li, Sarah Cogan, Jeremy Chen, Chao Jia, Chenjie Gu, Qiao Zhang, Jordan Grimstad, Ale Jakse Hartman, Xavier Garcia, Thanumalayan Sankaranarayana Pillai, Jacob Devlin, Michael Laskin, Diego de Las Casas, Dasha Valter, Connie Tao, Lorenzo Blanco, Adri Puigdomnech Badia, David Reitter, Mianna Chen, Jenny Brennan, Clara Rivera, Sergey Brin, Shariq Iqbal, Gabriela Surita, Jane Labanowski, Abhi Rao, Stephanie Winkler, Emilio Parisotto, Yiming Gu, Kate Olszewska, Ravi Addanki, Antoine Miech, Annie Louis, Denis Teplyashin, Geoff Brown, Elliot Catt, Jan Balaguer, Jackie Xiang, Pidong Wang, Zoe Ashwood, Anton Briukhov, Albert Webson, Sanjay Ganapathy, Smit Sanghavi, Ajay Kannan, Ming-Wei Chang, Axel Stjerngren, Josip Djolonga, Yuting Sun, Ankur Bapna, Matthew Aitchison, Pedram Pejman, Henryk Michalewski, Tianhe Yu, Cindy Wang, Juliette Love, Junwhan Ahn, Dawn Bloxwich, Kehang Han, Peter Humphreys, Thibault Sellam, James Bradbury, Varun Godbole, Sina Samangooei, Bogdan Damoc, Alex Kaskasoli, Sbastien M. R. Arnold, Vijay Vasudevan, Shubham Agrawal, Jason Riesa, Dmitry Lepikhin, Richard Tanburn, Srivatsan Srinivasan, Hyeontaek Lim, Sarah Hodgkinson, Pranav Shyam, Johan Ferret, Steven Hand, Ankush Garg, Tom Le Paine, Jian Li, Yujia Li, Minh Giang, Alexander Neitz, Zheer Abbas, Sarah York, Machel Reid, Elizabeth Cole, Aakanksha Chowdhery, Dipanjan Das, Dominika Rogoziska, Vitaliy Nikolaev, Pablo Sprechmann, Zachary Nado, Lukas Zilka, Flavien Prost, Luheng He, Marianne Monteiro, Gaurav Mishra, Chris Welty, Josh Newlan, Dawei Jia, Miltiadis Allamanis, Clara Huiyi Hu, Raoul de Liedekerke, Justin Gilmer, Carl Saroufim, Shruti Rijhwani, Shaobo Hou, Disha Shrivastava, Anirudh Baddepudi, Alex Goldin, Adnan Ozturk, Albin Cassirer, Yunhan Xu, Daniel Sohn, Devendra Sachan, Reinald Kim Amplayo, Craig Swanson, Dessie Petrova, Shashi Narayan, Arthur Guez, Siddhartha Brahma, Jessica Landon, Miteyan Patel, Ruizhe Zhao, Kevin Villela, Luyu Wang, Wenhao Jia, Matthew Rahtz, Mai Gimnez, Legg Yeung, James Keeling, Petko Georgiev, Diana Mincu, Boxi Wu, Salem Haykal, Rachel Saputro, Kiran Vodrahalli, James Qin, Zeynep Cankara, Abhanshu Sharma, Nick Fernando, Will Hawkins, Behnam Neyshabur, Solomon Kim, Adrian Hutter, Priyanka Agrawal, Alex Castro-Ros, George van den Driessche, Tao Wang, Fan Yang, Shuo yiin Chang, Paul Komarek, Ross McIlroy, Mario Lui, Guodong Zhang, Wael Farhan, Michael Sharman, Paul Natsev, Paul Michel, Yamini Bansal, Siyuan Qiao, Kris Cao, Siamak Shakeri, Christina Butterfield, Justin Chung, Paul Kishan Rubenstein, Shivani Agrawal, Arthur Mensch, Kedar Soparkar, Karel Lenc, Timothy Chung, Aedan Pope, Loren Maggiore, Jackie Kay, Priya Jhakra, Shibo Wang, Joshua Maynez, Mary Phuong, Taylor Tobin, Andrea Tacchetti, Maja Trebacz, Kevin Robinson, Yash Katariya, Sebastian Riedel, Paige Bailey, Kefan Xiao, Nimesh Ghelani, Lora Aroyo, Ambrose Slone, Neil Houlsby, Xuehan Xiong, Zhen Yang, Elena Gribovskaya, Jonas Adler, Matteo Wirth, Lisa Lee, Music Li, Thais Kagohara, Jay Pavagadhi, Sophie Bridgers, Anna Bortsova, Sanjay Ghemawat, Zafarali Ahmed, Tianqi Liu, Richard Powell, Vijay Bolina, Mariko Iinuma, Polina Zablotskaia, James Besley, Da-Woon Chung, Timothy Dozat, Ramona Comanescu, Xiance Si, Jeremy Greer, Guo-

long Su, Martin Polacek, Raphaël Lopez Kaufman, Simon Tokumine, Hexiang Hu, Elena Buchatskaya, Yingjie Miao, Mohamed Elhawaty, Aditya Siddhant, Nenad Tomasev, Jinwei Xing, Christina Greer, Helen Miller, Shereen Ashraf, Aurko Roy, Zizhao Zhang, Ada Ma, Angelos Filos, Milos Besta, Rory Blevins, Ted Klimenko, Chih-Kuan Yeh, Soravit Changpinyo, Jiaqi Mu, Oscar Chang, Mantas Pajarskas, Carrie Muir, Vered Cohen, Charline Le Lan, Krishna Haridasan, Amit Marathe, Steven Hansen, Sholto Douglas, Rajkumar Samuel, Mingqiu Wang, Sophia Austin, Chang Lan, Jiepu Jiang, Justin Chiu, Jaime Alonso Lorenzo, Lars Lowe Sjösund, Sébastien Cevey, Zach Gleicher, Thi Avrahami, Anudhyan Boral, Hansa Srinivasan, Vittorio Selo, Rhys May, Konstantinos Aisopos, Léonard Hussenot, Livio Baldini Soares, Kate Baumli, Michael B. Chang, Adrià Recasens, Ben Caine, Alexander Pritzel, Filip Pavetic, Fabio Pardo, Anita Gergely, Justin Frye, Vinay Ramasesh, Dan Horgan, Kartikeya Badola, Nora Kassner, Subhrajit Roy, Ethan Dyer, Víctor Campos Campos, Alex Tomala, Yunhao Tang, Dalia El Badawy, Elspeth White, Basil Mustafa, Oran Lang, Abhishek Jindal, Sharad Vikram, Zhitao Gong, Sergi Caelles, Ross Hemsley, Gregory Thornton, Fangxiaoyu Feng, Wojciech Stokowiec, Ce Zheng, Phoebe Thacker, Çağlar Ünlü, Zhishuai Zhang, Mohammad Saleh, James Svensson, Max Bileschi, Piyush Patil, Ankesh Anand, Roman Ring, Katerina Tsihlias, Arpi Vezer, Marco Selvi, Toby Shevlane, Mikel Rodriguez, Tom Kwiatkowski, Samira Daruki, Keran Rong, Allan Dafoe, Nicholas FitzGerald, Keren Gu-Lemberg, Mina Khan, Lisa Anne Hendricks, Marie Pellat, Vladimir Feinberg, James Cobon-Kerr, Tara Sainath, Maribeth Rauh, Sayed Hadi Hashemi, Richard Ives, Yana Hasson, Eric Noland, Yuan Cao, Nathan Byrd, Le Hou, Qingze Wang, Thibault Sottiaux, Michela Paganini, Jean-Baptiste Lespiau, Alexandre Moufarek, Samer Hassan, Kaushik Shivakumar, Joost van Amersfoort, Amol Mandhane, Pratik Joshi, Anirudh Goyal, Matthew Tung, Andrew Brock, Hannah Sheahan, Vedant Misra, Cheng Li, Nemanja Rakićević, Mostafa Dehghani, Fangyu Liu, Sid Mittal, Junhyuk Oh, Seb Noury, Eren Sezener, Fantine Huot, Matthew Lamm, Nicola De Cao, Charlie Chen, Sidharth Mudgal, Romina Stella, Kevin Brooks, Gautam Vasudevan, Chenxi Liu, Mainak Chaitin, Nivedita Melinkeri, Aaron Cohen, Venus Wang, Kristie Seymore, Sergey Zubkov, Rahul Goel, Summer Yue, Sai Krishnakumaran, Brian Albert, Nate Hurley, Motoki Sano, Anhad Mohananey, Jonah Joughin, Egor Filonov, Tomasz Kępa, Yomna Eldawy, Jiawern Lim, Rahul Rishi, Shirin Badiezadegan, Taylor Bos, Jerry Chang, Sanil Jain, Sri Gayatri Sundara Padmanabhan, Subha Puttagunta, Kalpesh Krishna, Leslie Baker, Norbert Kalb, Vamsi Bedapudi, Adam Kurzrok, Shuntong Lei, Anthony Yu, Oren Litvin, Xiang Zhou, Zhichun Wu, Sam Sobell, Andrea Siciliano, Alan Papir, Robby Neale, Jonas Bragagnolo, Tej Toor, Tina Chen, Valentin Anklin, Feiran Wang, Richie Feng, Milad Gholami, Kevin Ling, Lijuan Liu, Jules Walter, Hamid Moghaddam, Arun Kishore, Jakub Adamek, Tyler Mercado, Jonathan Mallinson, Siddhinita Wandekar, Stephen Cagle, Eran Ofek, Guillermo Garrido, Clemens Lombriser, Maksim Mukha, Botu Sun, Hafeezul Rahman Mohammad, Josip Matak, Yadi Qian, Vikas Peswani, Pawel Janus, Quan Yuan, Leif Schelin, Oana David, Ankur Garg, Yifan He, Oleksii Duzhyi, Anton Älgmyr, Timothée Lottaz, Qi Li, Vikas Yadav, Luyao Xu, Alex Chinien, Rakesh Shivanna, Aleksandr Chuklin, Josie Li, Carrie Spadine, Travis Wolfe, Kareem Mohamed, Subhabrata Das, Zihang Dai, Kyle He, Daniel von Dincklage, Shyam Upadhyay, Akanksha Maurya, Luyan Chi, Sebastian Krause, Khalid Salama, Pam G Rabinovitch, Pavan Kumar Reddy M, Aarush Selvan, Mikhail Dekhtyarev, Golnaz Ghiasi, Erdem Guven, Himanshu Gupta, Boyi Liu, Deepak Sharma, Idan Heimlich Shtacher, Shachi Paul, Oscar Akerlund, François-Xavier Aubet, Terry Huang, Chen Zhu, Eric Zhu, Elico Teixeira, Matthew Fritze, Francesco Bertolini, Liana-Eleonora Marinescu, Martin Bille, Dominik Paulus, Khyatti Gupta, Tejasi Latkar, Max Chang, Jason Sanders, Roopa Wilson, Xuewei Wu, Yi-Xuan Tan, Lam Nguyen Thiet, Tulsee Doshi, Sid Lall, Swaroop Mishra, Wanming Chen, Thang Luong, Seth Benjamin, Jasmine Lee, Ewa Andrejczuk, Dominik Rabiej, Vipul Ranjan, Krzysztof Styrz, Pengcheng Yin, Jon Simon, Malcolm Rose Harriott, Mudit Bansal, Alexei Robsky, Geoff Bacon, David Greene, Daniil Mirylenka, Chen Zhou, Obaid Sarvana, Abhimanyu Goyal, Samuel Andermatt, Patrick Siegler, Ben Horn, Assaf Israel, Francesco Pongetti, Chih-Wei "Louis" Chen, Marco Selvatici, Pedro Silva, Kathie Wang, Jackson Tolins, Kelvin Guu, Roey Yogeve, Xiaochen Cai, Alessandro Agostini, Maulik Shah, Hung Nguyen, Noah O'Donnaile, Sébastien Pereira, Linda Friso, Adam Stambler, Adam Kurzrok, Chenkai Kuang, Yan Romanikhin, Mark Geller, ZJ Yan, Kane Jang, Cheng-Chun Lee, Wojciech Fica, Eric Malmi, Qijun Tan, Dan Banica, Daniel Balle, Ryan Pham, Yanping Huang, Diana Avram, Hongzhi Shi, Jasjot Singh, Chris Hidey, Niharika Ahuja, Pranab Saxena, Dan Dooley, Srividya Pranavi Potharaju, Eileen O'Neill, Anand Gokulchandran, Ryan Foley, Kai Zhao, Mike Dusenberry, Yuan Liu, Pulkit Mehta, Ragha Kotikalapudi, Chalence Safranek-Shrader, Andrew Goodman, Joshua Kessinger, Eran Globen, Prateek Kolhar, Chris Gorgolewski, Ali Ibrahim, Yang Song, Ali Eichenbaum, Thomas Brovelli, Sahitya Potluri, Preethi Lahoti, Cip Baetu, Ali Ghorbani, Charles Chen, Andy Crawford, Shalini Pal, Mukund Sridhar, Petru Gurita, Asier Mujika, Igor Petrovski, Pierre-Louis Cedo, Chenmei Li, Shiyuan Chen, Niccolò Dal Santo, Siddharth Goyal, Jitesh Punjabi, Karthik Kappaganthu, Chester Kwak, Pallavi LV, Sarmishta Velury, Himadri Choudhury, Jamie Hall, Premal Shah, Ricardo Figueira, Matt Thomas, Minjie Lu, Ting Zhou, Chintu Kumar, Thomas Jurdi, Sharat Chikkerur, Yenai Ma, Adams Yu, Soo Kwak, Victor Åhdel, Sujeevan Rajayogam, Travis Choma, Fei Liu, Aditya Barua, Colin Ji, Ji Ho Park, Vincent Hellendoorn, Alex Bailey, Taylan Bilal, Huanjie Zhou, Mehrdad Khatir, Charles Sutton, Wojciech Rządowski, Fiona Mac-

intosh, Roopali Vij, Konstantin Shagin, Paul Medina, Chen Liang, Jinjing Zhou, Pararth Shah, Yingying Bi, Attila Dankovics, Shipra Banga, Sabine Lehmann, Marissa Bredesen, Zifan Lin, John Eric Hoffmann, Jonathan Lai, Raynald Chung, Kai Yang, Nihal Balani, Arthur Bražinskas, Andrei Sozanschi, Matthew Hayes, Héctor Fernández Alcalde, Peter Makarov, Will Chen, Antonio Stella, Liselotte Snijders, Michael Mandl, Ante Kärrman, Paweł Nowak, Xinyi Wu, Alex Dyck, Krishnan Vaidyanathan, Raghavender R, Jessica Mallet, Mitch Rudominer, Eric Johnston, Sushil Mittal, Akhil Udathu, Janara Christensen, Vishal Verma, Zach Irving, Andreas Santucci, Gamaleldin Elsayed, Elnaz Davoodi, Marin Georgiev, Ian Tenney, Nan Hua, Geoffrey Cideron, Edouard Leurent, Mahmoud Alnahlawi, Ionut Georgescu, Nan Wei, Ivy Zheng, Dylan Scandinaro, Heinrich Jiang, Jasper Snoek, Mukund Sundararajan, Xuezhi Wang, Zack Ontiveros, Itay Karo, Jeremy Cole, Vinu Rajashekhar, Lara Tumeh, Eyal Ben-David, Rishub Jain, Jonathan Uesato, Romina Datta, Oskar Bunyan, Shimu Wu, John Zhang, Piotr Stanczyk, Ye Zhang, David Steiner, Subhajit Naskar, Michael Azzam, Matthew Johnson, Adam Paszke, Chung-Cheng Chiu, Jaume Sanchez Elias, Afroz Mohiuddin, Faizan Muhammad, Jin Miao, Andrew Lee, Nino Vieillard, Jane Park, Jiageng Zhang, Jeff Stanway, Drew Garmon, Abhijit Karmarkar, Zhe Dong, Jong Lee, Aviral Kumar, Luwei Zhou, Jonathan Evens, William Isaac, Geoffrey Irving, Edward Loper, Michael Fink, Isha Arkatkar, Nanxin Chen, Izhak Shafraan, Ivan Petrychenko, Zhe Chen, Johnson Jia, Anselm Levskaya, Zhenkai Zhu, Peter Grabowski, Yu Mao, Alberto Magni, Kaisheng Yao, Javier Snaider, Norman Casagrande, Evan Palmer, Paul Suganthan, Alfonso Castaño, Irene Giannoumis, Wooyeol Kim, Mikołaj Rybiński, Ashwin Sreevatsa, Jennifer Prendki, David Soergel, Adrian Goedeckemeyer, Willi Gierke, Mohsen Jafari, Meenu Gaba, Jeremy Wiesner, Diana Gage Wright, Yawen Wei, Harsha Vashisht, Yana Kulizhskaya, Jay Hoover, Maigo Le, Lu Li, Chimezie Iwuanyanwu, Lu Liu, Kevin Ramirez, Andrey Khorlin, Albert Cui, Tian LIN, Marcus Wu, Riccardo Aguilar, Keith Pallo, Abhishek Chakladar, Ginger Perng, Elena Allica Abellan, Mingyang Zhang, Ishita Dasgupta, Nate Kushman, Ivo Penchev, Alena Repina, Xihui Wu, Tom van der Weide, Priya Ponnappalli, Caroline Kaplan, Jiri Simsa, Shuangfeng Li, Olivier Dousse, Fan Yang, Jeff Piper, Nathan Ie, Rama Pasumarthi, Nathan Lintz, Anitha Vijayakumar, Daniel Andor, Pedro Valenzuela, Minnie Lui, Cosmin Paduraru, Daiyi Peng, Katherine Lee, Shuyuan Zhang, Somer Greene, Duc Dung Nguyen, Paula Kurylowicz, Cassidy Hardin, Lucas Dixon, Lili Janzer, Kiam Choo, Ziqiang Feng, Biao Zhang, Achintya Singhal, Dayou Du, Dan McKinnon, Natasha Antropova, Tolga Bolukbasi, Orgad Keller, David Reid, Daniel Finchelstein, Maria Abi Raad, Remi Crocker, Peter Hawkins, Robert Dadashi, Colin Gaffney, Ken Franko, Anna Bulanova, Rémi Leblond, Shirley Chung, Harry Askham, Luis C. Cobo, Kelvin Xu, Felix Fischer, Jun Xu, Christina Sorokin, Chris Alberti, Chu-Cheng Lin, Colin Evans, Alek Dimitriev, Hannah Forbes, Dylan Banarse, Zora Tung, Mark Omernick, Colton Bishop, Rachel Sterneck, Rohan Jain, Jiawei Xia, Ehsan Amid, Francesco Piccinno, Xingyu Wang, Praseem Banzal, Daniel J. Mankowitz, Alex Polozov, Victoria Krakovna, Sasha Brown, MohammadHossein Bateni, Dennis Duan, Vlad Firoiu, Meghana Thotakuri, Tom Natan, Matthieu Geist, Ser tan Girgin, Hui Li, Jiayu Ye, Ofir Roval, Reiko Tojo, Michael Kwong, James Lee-Thorp, Christopher Yew, Danila Sinopalnikov, Sabela Ramos, John Mellor, Abhishek Sharma, Kathy Wu, David Miller, Nicolas Sonnerat, Denis Vnukov, Rory Greig, Jennifer Beattie, Emily Caveness, Libin Bai, Julian Eisenschlos, Alex Korchemniy, Tomy Tsai, Mimi Jasarevic, Weize Kong, Phuong Dao, Zeyu Zheng, Frederick Liu, Fan Yang, Rui Zhu, Tian Huey Teh, Jason Sanmiya, Evgeny Gladchenko, Nejc Trdin, Daniel Toyama, Evan Rosen, Sasan Tavakkol, Linting Xue, Chen Elkind, Oliver Woodman, John Carpenter, George Papamakarios, Rupert Kemp, Sushant Kafle, Tanya Grunina, Rishika Sinha, Alice Talbert, Diane Wu, Denese Owusu-Afriyie, Cosmo Du, Chloe Thornton, Jordi Pont-Tuset, Pradyumna Narayana, Jing Li, Saaber Fatehi, John Wieting, Omar Ajmeri, Benigno Uria, Yeongil Ko, Laura Knight, Amélie Héliou, Ning Niu, Shane Gu, Chenxi Pang, Yeqing Li, Nir Levine, Ariel Stolovich, Rebeca Santamaria-Fernandez, Sonam Goenka, Wenny Yustalim, Robin Strudel, Ali Elqursh, Charlie Deck, Hyo Lee, Zonglin Li, Kyle Levin, Raphael Hoffmann, Dan Holtmann-Rice, Olivier Bachem, Sho Arora, Christy Koh, Soheil Hassas Yeganeh, Siim Pöder, Mukarram Tariq, Yanhua Sun, Lucian Ionita, Mojtaba Seyedhosseini, Pouya Tafti, Zhiyu Liu, Anmol Gulati, Jasmine Liu, Xinyu Ye, Bart Chrzasczcz, Lily Wang, Nikhil Sethi, Tianrun Li, Ben Brown, Shreya Singh, Wei Fan, Aaron Parisi, Joe Stanton, Vinod Koverkathu, Christopher A. Choquette-Choo, Yunjie Li, TJ Lu, Abe Ittycheriah, Prakash Shroff, Mani Varadarajan, Sanaz Bahargam, Rob Willoughby, David Gaddy, Guillaume Desjardins, Marco Cornero, Brona Robenek, Bhavishya Mittal, Ben Albrecht, Ashish Shenoy, Fedor Moiseev, Henrik Jacobsson, Alireza Ghaffarkhah, Morgane Rivière, Alanna Walton, Clément Crepy, Alicia Parrish, Zongwei Zhou, Clement Farabet, Carey Radebaugh, Praveen Srinivasan, Claudia van der Salm, Andreas Fidjeland, Salvatore Scellato, Eri Latorre-Chimoto, Hanna Klimczak-Plucińska, David Bridson, Dario de Cesare, Tom Hudson, Piermaria Mendolicchio, Lexi Walker, Alex Morris, Matthew Mauger, Alexey Guseynov, Alison Reid, Seth Odoom, Lucia Loher, Victor Cotruta, Madhavi Yenugula, Dominik Grewe, Anastasia Petrushkina, Tom Duerig, Antonio Sanchez, Steve Yadlowsky, Amy Shen, Amir Globerson, Lynette Webb, Sahil Dua, Dong Li, Surya Bhupatiraju, Dan Hurt, Haroon Qureshi, Ananth Agarwal, Tomer Shani, Matan Eyal, Anuj Khare, Shreyas Rammohan Belle, Lei Wang, Chetan Tekur, Mihir Sanjay Kale, Jinliang Wei, Ruoxin Sang, Brennan Saeta, Tyler Liechty, Yi Sun, Yao Zhao, Stephan Lee, Pandu Nayak, Doug Fritz, Manish Reddy Vuyyuru, John Aslanides,

Nidhi Vyas, Martin Wicke, Xiao Ma, Evgenii Eltyshov, Nina Martin, Hardie Cate, James Manyika, Keyvan Amiri, Yelin Kim, Xi Xiong, Kai Kang, Florian Luisier, Nilesch Tripuraneni, David Madras, Mandy Guo, Austin Waters, Oliver Wang, Joshua Ainslie, Jason Baldridge, Han Zhang, Garima Pruthi, Jakob Bauer, Feng Yang, Riham Mansour, Jason Gelman, Yang Xu, George Polovets, Ji Liu, Honglong Cai, Warren Chen, XiangHai Sheng, Emily Xue, Sherjil Ozair, Christof Angermueller, Xiaowei Li, Anoop Sinha, Weiren Wang, Julia Wiesinger, Emmanouil Koukoumidis, Yuan Tian, Anand Iyer, Madhu Gurumurthy, Mark Goldenson, Parashar Shah, MK Blake, Hongkun Yu, Anthony Urbanowicz, Jennimaria Palomaki, Chrisantha Fernando, Ken Durden, Harsh Mehta, Nikola Momchev, Elahe Rahimtoroghi, Maria Georgaki, Amit Raul, Sebastian Ruder, Morgan Redshaw, Jinhyuk Lee, Denny Zhou, Komal Jalan, Dinghua Li, Blake Hechtman, Parker Schuh, Milad Nasr, Kieran Milan, Vladimir Mikulik, Juliana Franco, Tim Green, Nam Nguyen, Joe Kelley, Aroma Mahendru, Andrea Hu, Joshua Howland, Ben Vargas, Jeffrey Hui, Kshitij Bansal, Vikram Rao, Rakesh Ghiya, Emma Wang, Ke Ye, Jean Michel Sarr, Melanie Moranski Preston, Madeleine Elish, Steve Li, Aakash Kaku, Jigar Gupta, Ice Pasupat, Da-Cheng Juan, Milan Someswar, Tejvi M., Xinyun Chen, Aida Amini, Alex Fabrikant, Eric Chu, Xuanyi Dong, Amruta Muthal, Senaka Buthpitiya, Sarthak Jauhari, Nan Hua, Urvashi Khandelwal, Ayal Hitron, Jie Ren, Larissa Rinaldi, Shahar Drath, Avigail Dabush, Nan-Jiang Jiang, Harshal Godhia, Uli Sachs, Anthony Chen, Yicheng Fan, Hagai Taitelbaum, Hila Noga, Zhuyun Dai, James Wang, Chen Liang, Jenny Hamer, Chun-Sung Ferng, Chenel Elkind, Aviel Atias, Paulina Lee, Vít Listík, Mathias Carlen, Jan van de Kerkhof, Marcin Pikus, Krunoslav Zaher, Paul Müller, Sasha Zykova, Richard Stefanec, Vitaly Gatsko, Christoph Hirsenschall, Ashwin Sethi, Xingyu Federico Xu, Chetan Ahuja, Beth Tsai, Anca Stefanoiu, Bo Feng, Keshav Dhandhanian, Manish Katyal, Akshay Gupta, Atharva Parulekar, Divya Pitta, Jing Zhao, Vivaan Bhatia, Yashodha Bhavnani, Omar Alhadlaq, Xiaolin Li, Peter Danenberg, Dennis Tu, Alex Pine, Vera Filippova, Abhipso Ghosh, Ben Limonchik, Bhargava Urala, Chaitanya Krishna Lanka, Derik Clive, Yi Sun, Edward Li, Hao Wu, Kevin Hongtongsak, Ianna Li, Kalind Thakkar, Kuanysh Omarov, Kushal Majmundar, Michael Alverson, Michael Kucharski, Mohak Patel, Mudit Jain, Maksim Zabelin, Paolo Pelagatti, Rohan Kohli, Saurabh Kumar, Joseph Kim, Swetha Sankar, Vineet Shah, Lakshmi Ramachandruni, Xiangkai Zeng, Ben Bariach, Laura Weidinger, Tu Vu, Alek Andreev, Antoine He, Kevin Hui, Sheleem Kashem, Amar Subramanya, Sissie Hsiao, Demis Hassabis, Koray Kavukcuoglu, Adam Sadovsky, Quoc Le, Trevor Strohman, Yonghui Wu, Slav Petrov, Jeffrey Dean, and Oriol Vinyals. Gemini: A family of highly capable multimodal models, 2025.

- [73] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.
- [74] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.
- [75] Anders Tychsen, Doris McIlwain, Thea Brolund, and Michael Hitchens. Player-character dynamics in multi-player role-playing games. In *Proceedings of the 3rd Digital Games Research Association International Conference: “Situating Play”, DiGRA 2007*, pages 40–48. Digital Games Research Association (DiGRA), 2007.
- [76] G. Valcamonica, F. Piferi, V. Longo, N. Giallongo, F. Vona, and F. Garzotto. A gui framework for ai-generated cooperative interactive stories in educational settings. In *INTED2026 Proceedings*, 20th International Technology, Education and Development Conference, page 2216. IATED, 2-4 March, 2026 2026.
- [77] Arthur van der Torre. *Large language models and narrative storytelling in video games*. PhD thesis, 04 2025.
- [78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.
- [79] Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, Daniel Cer, Alice Lisak, Min Choi, Lucas Gonzalez, Omar Sanseviero, Glenn Cameron, Ian Ballantyne, Kat Black, Kaifeng Chen, Weiye Wang, Zhe Li, Gus Martins, Jinhyuk Lee, Mark Sherwood, Juyeong Ji, Renjie Wu, Jingxiao Zheng, Jyotinder Singh, Abheesh Sharma, Divyashree Sreepathihalli, Aashi Jain, Adham Elarabawy, AJ Co, Andreas Doumanoglou, Babak Samari, Ben Hora, Brian Potetz, Dahun Kim, Enrique Alfonseca, Fedor Moiseev, Feng Han, Frank Palma Gomez, Gustavo Hernández Ábrego, Heseng Zhang, Hui Hui, Jay Han, Karan Gill, Ke Chen, Koert Chen,

- Madhuri Shanbhogue, Michael Boratko, Paul Suganthan, Sai Meher Karthik Duddu, Sandeep Mariserla, Setareh Ariaifar, Shanfeng Zhang, Shijie Zhang, Simon Baumgartner, Sonam Goenka, Steve Qiu, Tanmaya Dabral, Trevor Walker, Vikram Rao, Waleed Khawaja, Wenlei Zhou, Xiaoqi Ren, Ye Xia, Yichang Chen, Yi-Ting Chen, Zhe Dong, Zhongli Ding, Francesco Visin, Gaël Liu, Jiageng Zhang, Kathleen Kenealy, Michelle Casbon, Ravin Kumar, Thomas Mesnard, Zach Gleicher, Cormac Brick, Olivier Lacombe, Adam Roberts, Qin Yin, Yunhsuan Sung, Raphael Hoffmann, Tris Warkentin, Armand Joulin, Tom Duerig, and Mojtaba Seyedhosseini. Embeddinggemma: Powerful and lightweight text representations, 2025.
- [80] Feng Wang, Yuqing Li, and Han Xiao. jina-reranker-v3: Last but not late interaction for listwise document reranking, 2025.
- [81] Yi Wang, Qian Zhou, and David Ledo. 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, 2024. Association for Computing Machinery.
- [82] Wizards of the Coast. *Dungeon Master's Guide*. Wizards of the Coast, Renton, WA, 5th edition, 2014.
- [83] Wizards of the Coast. *Player's Handbook*. Wizards of the Coast, Renton, WA, revised 5th edition (2024) edition, 2024.
- [84] Taewoo Yoo and Yun-Gyung Cheong. Leveraging llm-constructed graphs for effective goal-driven storytelling. In *LKM@IJCAI*, 2024.
- [85] José P. Zagal and Sebastian Deterding. Definitions of “role-playing games”. In José P. Zagal and Sebastian Deterding, editors, *Role-Playing Game Studies: Transmedia Foundations*, pages 19–53. Routledge, 2018.
- [86] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.
- [87] Mingzhe Zheng, Dingjie Song, Guanyu Zhou, Jun You, Jiahao Zhan, Xuran Ma, Xinyuan Song, Ser-Nam Lim, Qifeng Chen, and Harry Yang. CML-Bench: A framework for evaluating and enhancing LLM-powered movie scripts generation. *arXiv preprint*, arXiv:2510.06231, 2025.
- [88] Yufan Zhuang, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. Text generation beyond discrete token sampling, 2025.
- [89] Zilliz. What is annoy? <https://zilliz.com/learn/what-is-annoy>, 2024. Accessed: 2025-01-16.
- [90] Zilliz Learn. Top open-source vector search engines: A comprehensive comparison. [https://medium.com/@zilliz\\_learn/top-5-open-source-vector-search-engines-a-comprehensive-comparison-guide-for-2025-dd](https://medium.com/@zilliz_learn/top-5-open-source-vector-search-engines-a-comprehensive-comparison-guide-for-2025-dd) 2024. Accessed: 2025-01-16.

## A. User Questionnaire

In this section, the questionnaire administered to the users is presented and briefly described. For each section of the questionnaire, we show the questions, the range of possible answers, and the corresponding label used in the main text.

### A.1. User information

This section of the questionnaire is intended to collect information about the user’s familiarity with the main concepts behind the application (RPGs, LLMs, VINEs). The section is introduced by the following text:

*"Please provide the requested information."*

The information for this section is shown in Table 39.

Table 39: User information items

| Label                                    | Item                                                                                                                 | Answers                                         |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| RPG familiarity                          | What is your familiarity with role-playing games?                                                                    | 1 (Totally Unfamiliar) – 5 (Extremely familiar) |
| Generative AI (LLM) familiarity          | What is your familiarity with textual generative AI (LLMs)?                                                          | 1 (Totally Unfamiliar) – 5 (Extremely familiar) |
| Interactive narrative familiarity        | What is your familiarity with virtual interactive narrative experiences (narrative videogames, visual novels, etc.)? | 1 (Totally Unfamiliar) – 5 (Extremely familiar) |
| Game frequency (RPG/interactive fiction) | How often do you play role-playing games (e.g., D&D) or interactive fiction?                                         | Never, Rarely, Sometimes, Often, Very Often     |

## A.2. UEQ

This section of the questionnaire is intended to collect information about the user’s opinions on the usability of the application, using the short User Experience Questionnaire (UEQ). The section is introduced by the following text:

*"For each item, select the level that best describes your experience with the system between the two extremes."*

The information for this section is shown in Table 40.

Table 40: UEQ items and corresponding labels

| Label | Item / Answers                        |
|-------|---------------------------------------|
| UEQ1  | 1 (obstructive) – 7 (supportive)      |
| UEQ2  | 1 (complicated) – 7 (easy)            |
| UEQ3  | 1 (inefficient) – 7 (efficient)       |
| UEQ4  | 1 (confusing) – 7 (clear)             |
| UEQ5  | 1 (boring) – 7 (exciting)             |
| UEQ6  | 1 (not interesting) – 7 (interesting) |
| UEQ7  | 1 (conventional) – 7 (inventive)      |
| UEQ8  | 1 (usual) – 7 (leading edge)          |

## A.3. SUS

This section of the questionnaire is intended to collect information about the user’s opinions on the usability of the application, using the System Usability Scale (SUS). The section is introduced by the following text:

*"For each statement, select your agreement level between 1 (Strongly disagree) and 5 (Strongly agree)."*

The information for this section is shown in Table 41.

Table 41: SUS items and corresponding labels

| Label | Item                                                                                       | Answers                                    |
|-------|--------------------------------------------------------------------------------------------|--------------------------------------------|
| SUS1  | I think that I would like to use this system frequently.                                   | 1 (Strongly Disagree) – 5 (Strongly Agree) |
| SUS2  | I found the system unnecessarily complex.                                                  | 1 (Strongly Disagree) – 5 (Strongly Agree) |
| SUS3  | I thought the system was easy to use.                                                      | 1 (Strongly Disagree) – 5 (Strongly Agree) |
| SUS4  | I think that I would need the support of a technical person to be able to use this system. | 1 (Strongly Disagree) – 5 (Strongly Agree) |
| SUS5  | I found the various functions in this system were well integrated.                         | 1 (Strongly Disagree) – 5 (Strongly Agree) |
| SUS6  | I thought there was too much inconsistency in this system.                                 | 1 (Strongly Disagree) – 5 (Strongly Agree) |
| SUS7  | I would imagine that most people would learn to use this system very quickly.              | 1 (Strongly Disagree) – 5 (Strongly Agree) |
| SUS8  | I found the system very cumbersome to use.                                                 | 1 (Strongly Disagree) – 5 (Strongly Agree) |
| SUS9  | I felt very confident using the system.                                                    | 1 (Strongly Disagree) – 5 (Strongly Agree) |
| SUS10 | I needed to learn a lot of things before I could get going with this system.               | 1 (Strongly Disagree) – 5 (Strongly Agree) |

#### A.4. SE

This section of the questionnaire is intended to collect information about the user’s opinions on the quality of the generated stories and of the overall experience. The section is introduced by the following text:

*"The following questions explore your experience with the immersive storytelling task of the system."*

The information for this section is shown in Table 42.

Table 42: Story Evaluation items and corresponding labels

| Label | Item                                                                                                                                                                                   | Answers                                             |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------|
| SE1   | To what extent did you feel that the story reacted to your actions?                                                                                                                    | 1 (Very low) – 5 (Very high)                        |
| SE2   | On a scale from 1 (completely incoherent) to 5 (completely coherent), how would you rate the logical coherence of the story?                                                           | 1 (Completely incoherent) – 5 (Completely coherent) |
| SE3   | On a scale from 1 (completely incoherent) to 5 (completely coherent), how would you rate the coherence of the characters’ behaviour?                                                   | 1 (Completely incoherent) – 5 (Completely coherent) |
| SE4   | On a scale from 1 (completely unnatural) to 5 (completely natural), how would you rate the naturalness of the character reactions to your actions?                                     | 1 (Completely unnatural) – 5 (Completely natural)   |
| SE5   | On a scale from 1 (very shallow) to 5 (very evocative), how would you rate the evocativeness of the descriptions of locations and actions?                                             | 1 (Very shallow) – 5 (Very evocative)               |
| SE6   | On a scale from 1 (very low) to 5 (very high), how would you rate the comprehensibility and precision of the descriptions of locations and actions?                                    | 1 (Very low) – 5 (Very high)                        |
| SE7   | On a scale from 1 (totally unfitting) to 5 (totally fitting), how would you rate the appropriateness of the language used with respect to the fantasy/fairy-tale setting of the story? | 1 (Totally unfitting) – 5 (Totally fitting)         |
| SE8   | On a scale from 1 (very dull) to 5 (very creative), how would you rate the creativity of the story?                                                                                    | 1 (Very dull) – 5 (Very creative)                   |
| SE9   | Select your agreement level between 1 (strongly disagree) and 5 (strongly agree) with the following statement: My choices had a visible impact on how the story evolved.               | 1 (Strongly Disagree) – 5 (Strongly Agree)          |
| SE10  | Select your agreement level between 1 (strongly disagree) and 5 (strongly agree) with the following statement: I felt free to act in ways that mattered to the plot.                   | 1 (Strongly Disagree) – 5 (Strongly Agree)          |
| SE11  | Select your agreement level between 1 (strongly disagree) and 5 (strongly agree) with the following statement: The story gave me meaningful decisions to make.                         | 1 (Strongly Disagree) – 5 (Strongly Agree)          |
| SE12  | Select your agreement level between 1 (strongly disagree) and 5 (strongly agree) with the following statement: I was fully absorbed in the fantasy world.                              | 1 (Strongly Disagree) – 5 (Strongly Agree)          |
| SE13  | Select your agreement level between 1 (strongly disagree) and 5 (strongly agree) with the following statement: While interacting, I lost track of time.                                | 1 (Strongly Disagree) – 5 (Strongly Agree)          |
| SE14  | Select your agreement level between 1 (strongly disagree) and 5 (strongly agree) with the following statement: I felt as if I was inside the story.                                    | 1 (Strongly Disagree) – 5 (Strongly Agree)          |
| SE15  | How mentally demanding did you find the interaction?                                                                                                                                   | 1 (Very low) – 5 (Very high)                        |
| SE16  | In general, how would you rate the story on a scale from 1 to 5?                                                                                                                       | 1 – 5                                               |
| SE17  | In general, how would you rate the overall experience on a scale from 1 to 5?                                                                                                          | 1 – 5                                               |

## B. Prompts

In this section, we show and briefly comment on the prompts used for the various functions of the system. For each function, we present both the system prompt and the structure of the user prompt, with placeholders for the variable elements.

### B.1. Scene Generation

The system prompt for the generation of the narrative scene is shown in Listing 1.

### Listing 1: System prompt for the narrator model

```
1 You are a narrator for an interactive fantasy storytelling application. You will produce a
  scene given the information you will receive. Your role is similar to that of a game master
  in a role-playing game.
2
3 There are different types of scenes you might need to produce:
4 - Introductory scene: in this scene, the character, its objective, motivation, and the location
  of the story are introduced.
5 - Descriptive scene: you will receive an overview of a place consisting of a description of the
  place, the paths that depart from it, the characters that can be found there, and the
  possible interactable events. You must give a comprehensive description of all these
  elements, as if the character sees them for the first time. Prioritize completeness of
  information rather than detail level; be sure to mention all the elements you are given
  (paths, characters, and events). Do not assume any action taken by the protagonist in this
  scene; just describe to the player what the protagonist sees.
6 - Responsive scene: you will receive a list of actions the protagonist of the story wants to
  perform. Some actions might later be specified to have succeeded or failed. In this case,
  acknowledge and narrate the failures or successes, and also what the character obtains or
  achieves. You should describe how the actions are performed, how the environment responds,
  and how the actions succeed or fail. In these scenes, the player might ask a character to
  join them in their quest. In this case, if the entity belongs to the characters list, it
  will always accept and join the protagonist, while if it belongs to the events list, it
  will always refuse.
7 - Ending scene: this is the final scene of the story, where the protagonist and their
  companions take separate paths and the adventure ends.
8 - Transition scene: this scene marks the passage from one narrative act to the next. It should
  be similar to a descriptive scene, but more solemn and narrative. It should also recall the
  objective and motivation of the protagonist.
9
10 The prompt you will receive will have some important elements:
11 - PROTAGONIST: this is the protagonist of the story. You will always be given their name and a
  description, sometimes also their objective and motivation. The protagonist is the main
  character of the story, and the story revolves around them. Always consider that the story
  should unfold from their point of view, so never reveal things that the protagonist should
  not know. Use the information you have to shape the story, but narrate it only if it makes
  sense for the protagonist to know it.
12 - INVENTORY: a list of the protagonist's objects. Use it to track whether the protagonist can
  use a certain object to perform some actions. If the protagonist tries to use an object
  they do not possess, narrate that the protagonist searches for that object among their
  possessions but cannot find it.
13 - PARTY: these are the protagonist's companions. If this entry is empty, the protagonist is
  alone. Remember that you may include their reactions to the described situations, or
  sometimes the protagonist might address them directly, but ensure they remain in the
  background, supporting the protagonist. Their dialogue or actions should be brief and
  consistent with any personality traits provided in their description.
14 - CONTEXT: this is information about the lore of the world, the characters, or the situations
  you are describing. Always remember that this information is known to you but not
  necessarily to the protagonist, so always evaluate how much of it to provide in the
  narration. You will receive a score for the "knowledge" of the protagonist. Use it as an
  indication of how much history and lore they should know (0 = nothing, 10 = a lot).
15 - PREVIOUS SCENE: you will receive a summary of the previous scene, to guide the unfolding of
  the current one.
16 - STORY SO FAR: you will receive a summary of the story so far, to have an idea of the main
  events that have occurred.
17 - ACT: you will receive a number corresponding to the current act of the story. The story has
  three acts. Act 1 introduces the character and the situation; it ends with the protagonist
  reaching the main location of the story. Act 2 contains the rising action, and ends with
  the protagonist facing the main challenge of their quest, having their objective right in
  front of them. Act 3 consists of the climax of the story and the subsequent debrief, which
  leads to the end of the story. Use this to modulate pacing.
18
19 The narration should be in third person, even if narrated from the point of view of the
  protagonist. Use a descriptive, evocative style. Employ sensory details (sight, sound,
  smell) and a slightly formal, archaic tone reminiscent of classic fantasy literature, but
  keep sentences clear and avoid overly convoluted prose. Prioritize atmosphere and clarity.
  NEVER address the user directly; NEVER produce output that does not belong to the story;
```

NEVER acknowledge that you are an AI, a narrator, or that this is a story. Do not ask the user what they want to do next; never add any information other than the narration itself; do not ask rhetorical questions; simply narrate the scene based on the actions provided. A scene should be about 1400 characters or 250--350 words.

We will now briefly comment on the structure of the prompt:

- **line 1.** This line introduces the model to its role and provides general information about the application. It contains many key words, such as **interactive fantasy application** and **role-playing game**.
- **lines 3 - 8.** Here, the prompt describes to the model the various types of scenes it will generate, as discussed in Section 3.2.1.
- **lines 10 - 17.** These lines describe to the model all the main information it will receive upon generation.
- **line 19.** This line provides additional information and specifications about the lexical and stylistic characteristics of the generated scenes.

In Listing 2, we provide the structure of the user prompt, with placeholders for variable information.

Listing 2: Narrative prompt structure with placeholders

```
1 SCENE TYPE: [type of scene]
2
3 CONTEXT. Use these information to shape the story and the character's behaviour. DO NOT convey
  these information directly to the player as plain text, rather use character's dialogues
  and behaviours or environmental storytelling.
4
5 [retrieved informations]
6
7 STORY IN THIS NARRATIVE ACT
8
9 [summary of the story in the current narrative act]
10
11 LAST SCENE
12
13 [last scene of the narration (if any)]
14
15 LOCATION: [macro-location in which the narrative act takes place]
16
17 PROTAGONIST INFO
18
19 Name: [character's name]
20 Description: [character's description]
21 Objective: [character's objective]
22 Motivation: [character's motivation]
23
24 Knowledge: [the level of the characteristic "Knowledge" of the character]
25
26 PARTY:
27 [character's companions and their description]
28
29 EQUIPMENTS: [list of the Items in character's equipment and inventory]
30
31 ROOM ELEMENTS
32
33 Description: [description of the current room (location)]
34
35 Events: [the list of interactable events in the room and their descriptions]
36
37 Characters: [the list of interactable Companions (supportive characters) in the room and their
  descriptions]
38
39 The protagonist want to perform the following actions:
40
41 [list of actions the user wants the character to perform]
42
43 In particular, they:
44
45 [list of the formatted actions, with relative failure/success and prizes]
46
```

```

47 ACT: [number of the current act]
48
49 Remember:
50 reply with only the story, no additional elements;
51 always narrate in third person;
52 never address the user directly;
53 never comment the story in any way

```

We will now briefly comment on the structure of the prompt:

- line 1. This line contains the current scene type, in reference to the ones listed in the system prompt and discussed in Section 3.2.1.
- lines 3 - 5. Here, the prompt describes the context and how to use it. The context is the information retrieved by the retrieval system. Unfortunately, the model proved unable to correctly judge to what degree it should convey this information (for example, it would mention the name of a character even if the user's character should not know it).
- lines 7 - 13. The summary of the story in this narrative act (up to two scenes prior) and the full last scene, to maintain coherence and continuity in the story.
- lines 17 - 25. The information about the character, as discussed in Section ???. The Knowledge characteristic score is included to guide how much the model should convey through narration in terms of knowledge about the world.
- lines 27 - 30. Other information about the character: their Items and Companions (see Section 3.2.1).
- lines 32 - 38. Information about the current Room (see Section 3.2.2).
- lines 40 - 46. Information about the actions the character should perform, and their outcomes.
- lines 50 - 54. Reiteration of some important instructions for the generation of the story.

## B.2. Action extraction

The system prompt for the action extraction model is shown in Listing 3.

Listing 3: System prompt for the information extractor model

```

1 You are an information extractor. You will receive a text in which the actions that the user
  wants to perform are described.
2 The text may be in first or third person. If in third person, consider only the actions
  performed by the PROTAGONIST, which will be specified later.
3 You must return a JSON following the provided schema, which needs to contain the following
  information:
4 - An ordered list of all the actions performed by the character. This consists of a list of
  strings.
5 - A list of special actions, which consists of a list of objects with the following format:
6   {'action': string, 'target': string, 'information': string}
7 The 'action' field must take one of the following values:
8   - "fighting": each action that consists of engaging in physical combat with the target.
  Examples: "I attack with my sword", "I charge against the enemy", "throws a punch".
9   - "athletics": each action that consists of using physical abilities to overcome an
  obstacle. Examples: "I try to lift the rock", "I jump over the river", "break the door
  open".
10  - "surviving": each action that consists of using survival skills to overcome an obstacle
  or reach a goal. Examples: "I raise my shield to protect myself", "I use a medikit to cure
  my wounds", "lit a fire".
11  - "tinkering": each action that consists of carefully interacting with an object in order
  to repair or modify it with precision, manual skill, and finesse. Examples: "I pick the
  lock", "I disarm the trap", "repairs the crossbow".
12  - "talking": each action that involves interacting with a creature to reach a goal.
  Examples: "I threaten the bandits", "I try to calm the bear", "says to him: 'why don't you
  join me?'".
13  - "observing": each action that involves perceiving the world around using the senses.
  Examples: "I look closely at the chest", "I pay attention to any strange sound", "sniffs
  the air to smell if something is burning".
14  - "spellcasting": each action that involves the use of spells or magic in general.
  Examples: "I cast fireball", "I use a spell to fly", "shoots lightning from her fingers".
15  - "thinking": each action that involves using knowledge or intuition to find a solution to
  a problem. Examples: "I try to solve the riddle", "I translate the writings", "studies the
  magical object".

```

```

16 The 'target' is the target of the action. You will be provided with a list of POSSIBLE TARGETS,
    and you must choose from that list.
17 The 'information' field must contain the specific action the user specified in the text, e.g.,
    the passage from which you extracted the action. If the text contains direct dialogues,
    they must be reported in the 'information' field. Return at most one special action for
    each action.
18 Here is an example:
19   input:
20     PROTAGONIST: Reo
21     POSSIBLE TARGETS: Yorm the Swordmaster, Golgi the goblin, Axe
22     I want to approach the tall man with the sword, and tell him: "Hi, traveller, what is
    your name?"
23   output:
24     {
25       "action_list":
26       [
27         "approach the tall man with a sword",
28         "tell him: "Hi, traveller, what is your name?"
29       ],
30       "special_actions":
31       [
32         {
33           "action": "talking",
34           "target": "Yorm the Swordmaster",
35           "information": "tell him: "Hi, traveller, what is your name?"
36         }
37       ]
38     }

```

We will now briefly comment on the structure of the prompt:

- lines 1-2. These lines introduce the model to its role and provide general information about the input it will receive and how to interpret it.
- line 3. Introduces the description of the expected output.
- line 4. Describes the first output field: the list of actions the user wants the character to perform.
- line 5. Introduces the concept of special actions, described in Section 3.1.2.
- lines 6-15. Explain the content of the "action" field, which corresponds to the action types discussed in Section 3.2.1.
- lines 16-17. Explain the content of the fields "target" and "information" (see Section 3.1.2).
- lines 18-38. These lines show an example of interaction, adopting a one-shot learning approach for the task. The reader can also use this as an example of a prompt.

### B.3. Path taking

The system prompt for the path taking model is shown in Listing 4.

Listing 4: System prompt for the path taking model

```

1 You are an information extractor from text.
2 Your goal is to detect if a character physically takes one of the given paths.
3
4 Input:
5 - A list of paths, each with a unique numeric ID and a description.
6 - A list of actions performed by the character.
7
8 Definition of "taking a path":
9 - The character starts moving along, enters, or clearly commits to that path.
10 - Looking at, mentioning, thinking about, or approaching a path does NOT count.
11 - Negations ("does not take") do NOT count.
12
13 Rules:
14 - If the character takes one of the paths, return its numeric ID.
15 - If no path is taken (or only mentioned/considered), you must return 0.
16 - Return your reasoning in the reasoning field of the output JSON, and only the number and
    nothing else in the content field. No extra text, no punctuation except the number. Before
    returning a path, you must be completely sure that the character takes the path, and not

```

```
just mentions or observes it.
```

We will now briefly comment on the structure of the prompt:

- lines 1-2. These lines introduce the model to its role.
- lines 4-6. Explain the input that the model should expect.
- lines 8-11. Clearly define the meaning of "taking a path".
- lines 13-16. Explain in detail the rules the model must follow to return the correct output.

An example of input to the model is shown in Listing 5.

Listing 5: Example of input prompt for the path taking model

```
1 PATHS:
2   {
3     "1": "A dark path, going into the oldest part of the forest",
4     "2": "A path where mushrooms seem to emit some light",
5     "3": "A bright path, going to the youngest part of the forest"
6   }
7
8 SCENE:
9 After carefully thinking, I decide to take the path with the strange mushrooms.
```

## B.4. Companion joining detection

The system prompt for detecting whether a companion joins the protagonist is shown in Listing 6.

Listing 6: System prompt for companion joining detection

```
1 You are an information extractor from text.
2 Your goal is to detect if a character joins the protagonist in their quest.
3
4 Input:
5 - The protagonist name, a list of characters separated by commas, and a narrated scene.
6
7 Output: a boolean 'joined' that is true if at least one character joins the protagonist, and a
8         string 'target' that must contain the name of the character that joined the protagonist.
9
10 Rules:
11 - If at least one character on the list accepts to join the protagonist, physically follows
12   them, or makes it clear that they want to join the protagonist, then 'joined' is true and
13   'target' is the full name of that character as it appears in the Characters list. Beware:
14   just showing interest does not count; it must be clear from the text that the character
15   wants to follow the protagonist.
16 - If all the characters refuse, 'joined' is false and 'target' is an empty string.
17 - If the character that joins the protagonist does not belong to the list of characters
18   provided, 'joined' is false and 'target' is an empty string.
19 - Return your reasoning in the 'reasoning' field of the output JSON, and only the boolean and
20   the name of the character in the content field. No extra text, no punctuation except the
21   boolean and the name of the character.
```

We will now briefly comment on the structure of the prompt:

- lines 1-2. These lines introduce the model to its role.
- lines 4-5. Explain the input that the model should expect.
- line 7. Explain the output expected from the model.
- lines 9-13. Explain in detail the rules the model must follow to return the correct output.

An example of the input prompt for this component is shown in Listing 7.

Listing 7: Example input prompt for companion joining detection

```
1 PROTAGONIST: Dora
2
3 CHARACTERS: Golgi The Goblin, Irina The Scholar
4
5 SCENE:
6 Dora approached Irina, who was studying an ancient tome at the edge of the clearing. "Irina,
7   would you like to join us on our quest?" she asked hopefully. Irina looked up, closed the
8   book, and nodded. "I would be honored to accompany you," she replied softly. Golgi,
```

overhearing the conversation, shrugged and muttered something about keeping an eye on them, but did not explicitly agree to join.

## B.5. Scene summarization

The system prompt for the scene summarizer model is shown in Listing 8.

Listing 8: System prompt for the scene summarizer model

```
1 You are a story summarizer for a fantasy interactive narrative.
2
3 You will receive a single scene from a story. Your goal is to generate a brief, factual summary
  that captures only the elements necessary to reconstruct the story's plot.
4
5 Length: 1 to 3 sentences (maximum 50 words).
6
7 What to include (if present in the scene):
8 - Actions taken by the protagonist that change the situation
9 - Introduction of new characters, locations, or significant objects
10 - Decisions with consequences
11 - Revelations of lore or critical plot information
12 - Outcomes of conflicts or challenges
13
14 What to exclude:
15 - Ambient descriptions, internal monologues, minor dialogue, repetitive actions
16
17 Rules:
18 - Do NOT hallucinate. Only summarize what is explicitly in the scene.
19 - Do NOT repeat information already established in previous scenes unless it is essential to
  understand the current scene.
20 - Do NOT add meta-commentary (e.g., "In this scene...", "Here is the summary:").
21 - Output only the summary text as plain prose (not bullet points).
```

We will now briefly comment on the structure of the prompt:

- line 1. Introduces the model to its role.
- line 3. Explains the input that the model should expect.
- line 5. Explains the expected length of the output.
- lines 7-12. Explain the elements of the scene on which the summary should focus.
- lines 14-15. Explain the elements of the scene that the summary should ignore.
- lines 17-21. List some important rules to respect during the summarization process.

## B.6. Generation without architecture

The system prompt for the generation model when used without the architecture is shown in Listing 9.

Listing 9: System prompt for generation without architecture

```
1 You are a narrator for an interactive fantasy storytelling application. You will produce a
  scene given the information you will receive. Your role is similar to that of a game master
  in a role-playing game.
2
3 The prompt you will receive will have some important elements:
4 - STORY OUTLINE: it contains the outline of the whole story from start to end. Use this
  information to guide the generation of the current scene. The current scene should not
  narrate the whole story, but just a scene which describes the current situation or reacts
  to the protagonist's actions.
5 - PROTAGONIST: this is the protagonist of the story. You will always be given their name and a
  description, sometimes also their objective and motivation. The protagonist is the main
  character of the story, and the story revolves around them. Always consider that the story
  should unfold from their point of view, so never reveal things that the protagonist should
  not know. Use the information you have to shape the story, but narrate it only if it makes
  sense for the protagonist to know it.
6 - EQUIPMENT: a list of the protagonist's objects. Use it to track whether the protagonist can
  use a certain object to perform some actions. If the protagonist tries to use an object
```

they do not possess, narrate that the protagonist searches for that object among their possessions but cannot find it. You will receive a score for the "knowledge" of the protagonist. Use it as an indication of how much history and lore they should know (0 = nothing, 10 = a lot).

- 7 - PREVIOUS SCENES: you will receive the previous scenes, to guide the unfolding of the current one.
  - 8 - The protagonist's actions: you will receive the user input describing what the protagonist wants to do in the current scene. You should narrate the actions and their consequences, and how the environment reacts to them.
  - 9 - Scene number: you will receive the number of the current scene compared with the maximum number of scenes of the story in the form [current scene number]/[max scenes in the story]. Use this information to shape the pace of the story. In particular, if the scene number is 1, the scene you must generate is introductory, and should introduce the character, their goal, and their motivation, as well as the starting location. If instead the current scene number is equal to the maximum number of scenes, this is the ending scene and you should end the story with the protagonist achieving their goal. Keep in mind that the narration should end with the protagonist obtaining their objective within the number of scenes indicated, so pace the story accordingly.
- 10
- 11 The narration should be in third person, even if narrated from the point of view of the protagonist. Use a descriptive, evocative style. Employ sensory details (sight, sound, smell) and a slightly formal, archaic tone reminiscent of classic fantasy literature, but keep sentences clear and avoid overly convoluted prose. Prioritize atmosphere and clarity. NEVER address the user directly; NEVER produce output that does not belong to the story; NEVER acknowledge that you are an AI, a narrator, or that this is a story. Do not ask the user what they want to do next; never add any information other than the narration itself; do not ask rhetorical questions; simply narrate the scene based on the actions provided. A scene should be about 1400 characters or 250--350 words.

We will now briefly comment on the structure of the prompt, also by comparing it with the prompt shown in Listing 1:

- line 1. This line introduces the model to its role and provides general information about the application. It contains many key words, such as **interactive fantasy application** and **role-playing game**. It is identical to the corresponding line in the prompt used with the architecture.
- lines 3 - 9. These lines describe to the model all the main information it will receive upon generation. They are similar to lines 10 - 17 of the prompt used with the architecture, but they differ in the contents and the description of some elements. The key differences are the presence of the natural language story outline and the **Scene number** element, which substitutes the **Act** as a mechanism to produce a narrative structure in the story.
- line 11. This line provides additional information and specifications about the lexical and stylistic characteristics of the generated scenes. It is identical to line 19 from the prompt used by the architecture.

In Listing 10, we provide the structure of the user prompt for the bypass, with placeholders for variable information.

Listing 10: Narrative prompt structure with placeholders for the bypass

```

1 SCENE TYPE: [type of scene]
2
3 STORY OUTLINE:
4
5 [natural language high level story outline]
6
7 PROTAGONIST INFO
8
9 Name: [character's name]
10 Description: [character's description]
11 Objective: [character's objective]
12 Motivation: [character's motivation]
13
14 Knowledge: [the level of the characteristic "Knowledge" of the character]
15
16 PARTY:
17 [character's companions and their description]
18
19 EQUIPMENTS: [list of the Items in character's equipment and inventory]
20
```

```
21 STORY SO FAR
22
23 [all the scenes until the last one produced (if any)]
24
25 The protagonist wants to perform the following actions:
26
27 [user input]
28
29 This is scene [current scene number]/[max scenes]
30
31 Remember:
32 reply with only the story, no additional elements;
33 always narrate in third person;
34 never address the user directly;
35 never comment the story in any way
```

## C. Story Outline

### C.1. Outline correspondence

In this section, we show the story outline in natural language and compare it with its version in formatted language. The limitations of this approach and their consequences on this work are discussed in Sections 3.4.2 and 7.1. Here, we restrict ourselves to discussing the outline itself, and what it accomplishes and misses in testing the architecture’s capabilities. Let us now analyze, act by act, the two outlines. Figure 23 shows the first act in both natural language and formatted form.

## Natural Language (Act 1)

The character starts in the Ancient Forest, a great and old forest, filled with magic and mystery, Here, on a small clearing, the character sees two people: a small green Goblin with red hair and a tuft of hair and which wears a worn-out leather armor, his name is Golgi, and a young girl with scholar robes and a wizard hat, with dark skin and curly hair, her name is Irina. Moreover, here the protagonist sees an axe lodged in a tree, that could be removed with enough strength. If the protagonist moves further into the forest, he arrives at the Ruined Castle, his destination.

## JSON (Act 1)

```
{
  "acts": [
    {
      "number": 1,
      "location": "Ancient Forest",
      "introductive_prompt": "The
        protagonist arrive to the
        borders of the Ancient Forest.
        The Ruined Castle, their
        destination, lays at heart of
        this forest. The protagonist
        must traverse this old and
        dangerous forest, of which
        they heard only through
        legends. The protagonist take
        a deep breath, then enters the
        Forest.",
      "rooms": [
        {
          "id": 1,
          "description": "A small
            clearing in the middle of
            the forest.",
          "characters": ["Golgi The
            Goblin", "Irina The
            Scholar"],
          "events": ["Axe"],
          "paths": ["A wide path
            leading forward, in the
            depths of the forest"],
          "connections": [1000]
        }
      ]
    }
  ]
}
```

Figure 23: (a) Act 1: natural language and JSON outline.

In this case, the first act is composed of a single location (room). In this location, two characters (Companions in our application) are presented: Irina the Scholar and Golgi the Goblin. Both are visible in the "characters" field of the formatted scene. Similarly, the axe lodged in the tree is reported as "Axe" in the "events" field. As can be seen, no significant amount of additional information is added in the formatted outline. Conversely, it seems from this comparison that the formatted input has lost information. That is, however, not the case. Additional information is stored in other fields of the formatted outlines, which will be discussed later. The value 1000 in the "connections" field indicates that the corresponding path described in the "paths" field leads to the end of the first act, as described in Section 3.2.2.

Figure 24 shows the second act.

## Natural Language (Act 2)

Here they enter the throne hall. In the throne hall they see a girl in shining armor, wielding a halberd, with orange hair arranged in two braids, her name is Miriam. In the same room there is a locked chest, with a golden lock. It will be very difficult to break it open or open it by sheer force, a little easier to use a spell to open it and not too difficult to pick the lock, but this action would be easier to do with the right key. If opened, the chest reveals to contain the Sword of Ruin and a Spell Scroll. If the protagonist proceeds deeper inside the dungeon, they will find themselves in a great crumbled hall. In this Hall there is a big knight in a black armor, wielding a big heavy shield and a greatsword, named Alken. He will attack the protagonist relentlessly, without spelling a single word. At first he can be talked to, or some tricks can be used, but in the end he must be defeated either by fighting him or using a spell on him. He can be defeated, and in this case his Black Iron Shield and a golden Magical Key that he was carrying can be taken. This key can open the chest seen before. Behind Alken there is a gigantic door. If the protagonist open it and go inside, they will find themselves in the deeper part of the castle.

## JSON (Act 2)

```
...
{
  "number": 2,
  "location": "Ruined Castle",
  "introductive_prompt": "The
    protagonist exit from the
    Ancient Forest, arriving in
    the valley at its heart, and
    now find themselves in front
    of the wide open gates of the
    Ruined Castle.",
  "rooms": [
    {
      "id": 1,
      "description": "The throne
        hall, big and sumptuous,
        once would have been
        splendid, now is dusty and
        ruined.",
      "characters": ["Miriam The
        Warrior"],
      "events": ["The Chest"],
      "paths": ["Stairs to the
        left, leading deep down in
        the castle dungeon, the
        path to move forward"],
      "connections": [7]
    },
    {
      "id": 7,
      "description": "A wide hall,
        once beautiful and
        sumptuous, now crumbling",
      "characters": [],
      "events": ["Alken the Mad
        Knight"],
      "paths": ["Ascending stairs,
        leading to the throne
        hall", "A gigantic door,
        that seems to lead to the
        deepest, oldest part of
        the castle. The
        protagonist objective
        probably lies on the other
        side of the door. the path
        to move forward"],
      "connections": [1, 1000]
    }
  ]
},
...
```

Figure 24: (b) Act 2: natural language and JSON outline.

### Natural Language (Act 3)

1. If they were searching for the Mystical Crown, they will encounter the Fallen King, an ancient immortal King who has now fallen into sorrow and despair. He is found in a collapsed room which overlooks a ravine. At the center of the room there is a grave, surrounded by flowers. He is sitting on the verge of a ravine, in a collapsed room, staring at the void. He has the crown, and can be convinced to give it up by talking, fighting or spellcasting. He will be unbothered by the attacks or the spells of the protagonist, but he will admire their determination. After giving up the crown, he will sit again at the verge of the ravine, and will be unresponsive.
2. If the protagonist is looking for the Treasure of the Fallen King, they will face the fearsome Dragon of Ruin, which guards the treasure. The treasure is inside a Ravine which is full of riches. When the protagonist enters the ravine, the mighty Dragon will land in front of him. Here the dragon can be faced with magic or fighting skills, but also with some clever tricks, elaborated traps or some talking. After the first interaction, the dragon will throw flame at the protagonist, which can be resisted, dodged or defended against with magic. Finally, the dragon must be defeated either physically or with magic. If defeated the dragon will retreat, leaving the treasure to be taken.
3. If the protagonist wants to defeat the Shadowmancer, they will face the powerful wizard, which will summon shadow monsters to fight them. After defeating the army, the protagonist faces the Shadowmancer himself. It can be defeated using magic or by fighting him, but also clever tricks or elaborated traps.

### JSON (Acts 3)

```
...
{
  "number": 3,
  "location": "Ruined Castle depths",
  "introductive_prompt": "The protagonist has
    faced many perils, and is now arrived to
    the deepest part of the Ruined Castle.
    After wondering within the desolated halls
    of the inner castle, the protagonist spot
    a large old door, half opened. The
    protagonist can't see inside the room, but
    they have the strong feeling that their
    objective lies behind that door.",
  "rooms": [
    {
      "id": 1,
      "description": "A big, once luxurious
        room. It looks like a memorial room.
        At the center of the room there is
        what appear to be grave, with no name
        on it, just adorned with beautiful
        flower. The opposite side of the room
        is torn, and it overlooks an enormous
        chasm.",
      "characters": [],
      "events": ["the fallen King"],
      "paths": ["The way to go back, leave this
        place and end the adventure."],
      "connections": [1000]
    },
    {
      "id": 2,
      "description": "A gigantic chasm,
        completely filled with treasures of
        all kind. There are golden coins,
        shimmering gems, incredible artefacts.
        All there accumulated and dusty, the
        treasure still shines under the sun
        rays, coming from the collapsed roof
        of what once was the treasure hall of
        the Ruined Castle",
      "characters": [],
      "events": ["Dragon of Ruins"],
      "paths": ["The way to go back, leave this
        place and end the adventure."],
      "connections": [1000]
    },
    {
      "id": 3,
      "description": "A wide rooftop of the
        tallest tower of the castle, where the
        whole valley around can be seen.",
      "characters": [],
      "events": ["Shadowmancer's Army"],
      "paths": ["The way to go back, leave this
        place and end the adventure."],
      "connections": [1000]
    }
  ]
},
...
```

Figure 25: (c) Act 3: natural language and JSON outline.

Act 2 is composed of two locations: the throne hall, featuring the character Miriam the Warrior and a Chest containing some treasure, and a crumbling great hall, where Alken is found, a big knight in black armor who will attack the protagonist. One noticeable peculiarity is that Alken is not listed among the **"characters"** but among the **"events"**. This is because Alken, although a character in the story, plays the role of an obstacle that the protagonist must overcome, not the role of a Companion who can help the protagonist in their journey. Another important detail to highlight is that the key that can be taken from Alken after he has been defeated can be used to open the chest in the previous location. This might seem not to be conveyed by the formatted outline. However, as already said, this information is stored elsewhere in the formatted outline.

Figure 25 shows the third act. In this case, the story outline (and the corresponding room) is chosen according to the goal selected by the user. For the non-architecture case, this selection is performed by the LLM, while in the case of the architecture, it is the architecture (or, more specifically, the application) that performs this selection. The third act is unique in that it branches into three distinct paths depending on the user's goal. This branching structure is represented in the JSON by three separate rooms, each with its own events and connections. This design choice allows the architecture to handle multiple narrative outcomes while maintaining a consistent underlying structure.

| Natural Language (Ending)                         | JSON (Ending)                                                                                                                                                                                                                                 |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| At the end the protagonist achieve<br>their goal. | <pre> ... {   "number": 4,   "location": "The Way Home",   "introductive_prompt": "The     Protagonist has achieved their     goal and ended their     adventure. He leave the Ruined     Castle, to return home.",   "rooms": [] } ] }</pre> |

Figure 26: (d) Ending: natural language and JSON outline.

Figure 26 shows the ending. This is simply the conclusion of the story, formatted as a fictional act.

## C.2. Additional elements

In addition to the act and room structure, the formatted outline includes several supporting elements that are essential for the application's functionality. These include Items, Events, and Companions, which are defined as separate JSON objects within the same file. The following tables present these elements in a more readable format.

Other than the acts and rooms previously presented, the formatted story outline contains additional information, which consists in both narrative and parametric elements for the functioning of the application. These data are saved as JSON objects, in the same file as the previously shown JSON **acts**. However, they will be represented in tabular form for simplicity.

The list of Items contained in the formatted outline is available in Table 43.

The first eight Items (until `silk_gloves`) are the "default" ones, belonging to the starting Equipment of the four Archetypes (See Section 3). The rest are Items obtainable in the adventure, and the reader might draw a correspondence with the objects mentioned in the natural language story outline.

The reader can compare the information in the table with the fields of the class Item in Section 3.2.

Table 43: List of equipment items defined in the story

| Name                   | Description                                                                                                         | Compatibilities                  | Characteristic | Bonus |
|------------------------|---------------------------------------------------------------------------------------------------------------------|----------------------------------|----------------|-------|
| sword                  | A simple straight sword, useful in combat.                                                                          | right_hand, left_hand            | strength       | 1     |
| shield                 | A simple round shield, good for self defense.                                                                       | right_hand, left_hand            | resistance     | 1     |
| leather_boots          | Robust leather boots, perfect for exploration in the wild.                                                          | armor                            | agility        | 1     |
| rope                   | An old but still functioning rope, can be used in multiple creative ways.                                           | right_hand, left_hand, accessory | instinct       | 1     |
| book_of_knowledge      | A book containing ancient knowledge, which most consider to be lost.                                                | right_hand, left_hand, accessory | knowledge      | 1     |
| necklace_of_influence  | An enchanted talisman, which makes the wearer appear more charming.                                                 | accessory                        | charisma       | 1     |
| violin                 | A well made violin, produces a clear and sweet sound.                                                               | right_hand, left_hand            | charisma       | 1     |
| silk_gloves            | Silky, delicate gloves, helps improving the manual skills of the wearer.                                            | armor, accessory                 | agility        | 1     |
| axe                    | A simple lumberjack axe, can be used in close combat.                                                               | right_hand, left_hand            | strength       | 2     |
| sword_of_ruin          | A fine crafted sword, with runes encrypted on its blade. It seems surrounded by a blue shimmering.                  | right_hand, left_hand            | strength       | 3     |
| spell_scroll           | A scroll containing formulas for ancient spells.                                                                    | right_hand, left_hand, accessory | knowledge      | 3     |
| magical_key            | A golden key, of excellent elegance. It emanates a mysterious magical aura. Probably opens something in the castle. | accessory                        | instinct       | 3     |
| black_iron_greatshield | A black, massive iron shield. It's extremely heavy, but offers absolute defense.                                    | right_hand, left_hand            | resistance     | 3     |

Tables 44 and 45 show the data about the Events. Again, these can be compared with the fields of the class Event discussed in Section 3.2. Some of these Events lead to other Events when overcome. This is reflected, in the natural language story outline, by the description of the events and how they play out. In the hypothesis of an automatic generation of the formatted outline, the model should be instructed to recognize longer and more complex interactions, and hence split these events into sub-events.

While the metadata table provides an overview of each event's role and structure, the difficulty table details the specific ability checks required to overcome them, along with the corresponding rewards.

An additional note can be made on the difficulties of the event **The Chest**, where **tinkering** gets a bonus from the **magical\_key** Item, following the mechanism discussed in Section 3.2, in the Formatted Action class. This reflects the indication of the natural language story outline that the key obtained by defeating Alken can help in opening the chest mentioned before. This example illustrates a key feature of the architecture: the ability to represent dependencies between events that span multiple rooms. In the natural language outline, this connection is described narratively; in the formatted outline, it is encoded through the **bonus\_object** field. This allows the application to automatically track and apply such dependencies during gameplay.

Table 44: Events defined in the story - Metadata

| Name                            | Description                                                                                                                                                                                                                                                                                                                                                                                     | Following Event                 | Block Path |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|------------|
| Axe                             | An axe stuck in a tree. It seems to be stuck very deeply, but with enough strength or with some kind of strategy could be pulled out.                                                                                                                                                                                                                                                           | –                               | –          |
| The Chest                       | A luxurious, elegant chest. It has a shining gold lock, which emanates a somewhat magical golden aura.                                                                                                                                                                                                                                                                                          | –                               | –          |
| Alken the Mad Knight            | An incredible tall and mighty knight, wearing a heavy black armor. He wields an immense shield and a gigantic sword. He immediately attacks the protagonist at sight. There is no escape from him.                                                                                                                                                                                              | Alken The Mad Knight            | True       |
| Alken The Mad Knight            | The enormous knight is still fighting. However, it seems that he is wavering. It can be defeated.                                                                                                                                                                                                                                                                                               | Alken defeated                  | True       |
| Alken defeated                  | Alken has been knocked unconscious by the protagonist’s action. He lies on the ground, unresponsive                                                                                                                                                                                                                                                                                             | –                               | –          |
| the fallen King                 | An old man is sitting by the verge of the chasm. He is staring at the abyss below, with a gloomy expression. He wears a refined armor, now old and rusty. And on his head stands the Mystical Crown.                                                                                                                                                                                            | the Fallen King                 | True       |
| the Fallen King                 | The Protagonist’s action seems to have caught the attention of the King. He now stands up, in all his might, in front of the Protagonist. The King challenges the Protagonist to convince him, one way or the other, to give up the Crown.                                                                                                                                                      | The Fallen King                 | True       |
| The Fallen King                 | The King has again started to stare into the abyss beneath him, seated on the torn side of the room. He is now unresponsive to any action of the Protagonist.                                                                                                                                                                                                                                   | –                               | –          |
| Dragon of Ruins                 | While the Protagonist and their companion are staring at the majestic treasure, a terrifying roar shakes the earth. A colossal flying dragon lands on top of the treasure, and lays its dreadful gaze upon the Protagonist and their party, waiting for their move.                                                                                                                             | The Flame                       | True       |
| The Flame                       | The Dragon breathes in, and then releases a terrible burst of flame from its mouth onto the Protagonist. The Protagonist sees the flame, and will have to act.                                                                                                                                                                                                                                  | The Dragon of Ruins             | True       |
| The Dragon of Ruins             | Once again the dragon advances, staring directly in the Protagonist’s eyes, waiting for their next move                                                                                                                                                                                                                                                                                         | The Treasure of the Fallen King | True       |
| The Treasure of the Fallen King | The treasure rests now unguarded, waiting for the Protagonist and their companions to take it.                                                                                                                                                                                                                                                                                                  | –                               | –          |
| Shadowmancer’s Army             | Curved on a black, large cauldron, the Shadowmancer is chanting obscure words, in order to cast his ultimate spell, that will plunge the world in darkness. When he notices the Protagonist and their companion, he is irritated, and summons an army of shadow monsters to fight the intruders, while he continues his ritual. The army blocks the Protagonist from reaching the Shadowmancer. | The Shadowmancer                | True       |
| The Shadowmancer                | The Shadowmancer appears irritated by the defeat of his army, and now attacks the Protagonist and their group directly, using his powerful Shadow Spells.                                                                                                                                                                                                                                       | –                               | True       |

Table 45: Events defined in the story - Difficulty checks

| Event Name           | Ability      | DC | Prizes                                                                                                                                            |
|----------------------|--------------|----|---------------------------------------------------------------------------------------------------------------------------------------------------|
| Axe                  | athletics    | 5  | axe                                                                                                                                               |
| Axe                  | spellcasting | 6  | axe                                                                                                                                               |
| Axe                  | fighting     | 7  | axe                                                                                                                                               |
| Axe                  | tinkering    | 7  | axe                                                                                                                                               |
| Axe                  | observing    | 3  | the character confirms that, in fact, it is just an axe stuck in a tree                                                                           |
| The Chest            | athletics    | 12 | spell_scroll, sword_of_ruin                                                                                                                       |
| The Chest            | tinkering    | 7  | spell_scroll, sword_of_ruin (bonus: magical_key)                                                                                                  |
| The Chest            | fighting     | 12 | spell_scroll, sword_of_ruin                                                                                                                       |
| The Chest            | spellcasting | 8  | spell_scroll, sword_of_ruin                                                                                                                       |
| Alken the Mad Knight | talking      | 9  | Alken seems to have listened to the protagonist for a moment, but then he attacks again, like a fury                                              |
| Alken the Mad Knight | fighting     | 7  | Alken seems to have accused the protagonist's attack but then he raises again and attacks, like a fury                                            |
| Alken the Mad Knight | spellcasting | 7  | Alken seems to have accused the protagonist's spell but then he raises again and attacks, like a fury                                             |
| Alken the Mad Knight | thinking     | 7  | The protagonist outsmarts Alken, causing him damage and confusion, but then he raises again and attacks, like a fury                              |
| Alken the Mad Knight | tinkering    | 7  | The protagonist's action has effect on Alken but then he raises again and attacks, like a fury                                                    |
| Alken The Mad Knight | fighting     | 5  | black_iron_greatshield, magical_key                                                                                                               |
| Alken The Mad Knight | spellcasting | 5  | black_iron_greatshield, magical_key                                                                                                               |
| the fallen King      | talking      | 6  | The King seems little interested in the Protagonist's word. However, he turns his gaze upon the protagonist. His expression betrays sorrow.       |
| the fallen King      | fighting     | 8  | The King is not even scratched by the Protagonist's attack. However, it seems that the Protagonist's determination has gotten his interest.       |
| the fallen King      | spellcasting | 8  | The King is not even scratched by the Protagonist's spells. However, it seems that the Protagonist's determination has gotten his interest.       |
| the Fallen King      | talking      | 4  | The Protagonist's words resonate with the King. He smiles for the first time in many years. He takes the Crown and hands it to the Protagonist... |
| the Fallen King      | fighting     | 8  | The insistence of the attacks of the Protagonist allows them to manage to scratch the King a bit...                                               |
| the Fallen King      | spellcasting | 8  | The King is not even scratched by the Protagonist's spells. However, it seems that the Protagonist's determination has gotten his interest.       |
| Dragon of Ruins      | talking      | 9  | The Protagonist's words seem to have a certain effect on the dragon. However, his gaze does not seem less threatening.                            |
| Dragon of Ruins      | fighting     | 7  | The Protagonist manages to damage the dragon with their fighting skill. The creature retreats briefly away from the Protagonist's attacks...      |
| Dragon of Ruins      | spellcasting | 8  | The Protagonist's spell seems to affect the dragon. However, when the creature rises its head, its eyes are burning with fury.                    |
| Dragon of Ruins      | thinking     | 7  | The protagonist outsmarts the Dragon, causing him damage and confusion.                                                                           |
| Dragon of Ruins      | tinkering    | 7  | The protagonist's action has effect on the Dragon, causing damage.                                                                                |
| The Flame            | athletics    | 5  | The Protagonist eludes the flame completely, and remains undamaged                                                                                |
| The Flame            | surviving    | 5  | The Protagonist resists the flame completely, and remains undamaged                                                                               |
| The Flame            | spellcasting | 5  | The Protagonist uses a spell to defend against the flame, and remains undamaged.                                                                  |
| The Dragon of Ruins  | talking      | 12 | The Protagonist's words seem to have affected the Dragon. The creature seems almost to bow in front of the Protagonist...                         |
| The Dragon of Ruins  | fighting     | 6  | The Protagonist's fighting skill overwhelms the Dragon, which retreats wounded...                                                                 |
| The Dragon of Ruins  | spellcasting | 7  | The Protagonist's powerful spell overwhelms the Dragon, which succumbs to its effect...                                                           |
| Shadowmancer's Army  | fighting     | 6  | The Protagonist defeats the shadow monsters                                                                                                       |
| Shadowmancer's Army  | spellcasting | 6  | The Protagonist defeats the shadow monsters                                                                                                       |
| The Shadowmancer     | fighting     | 7  | The Shadowmancer is defeated, his shadowy form begins to crumble...                                                                               |
| The Shadowmancer     | spellcasting | 7  | The Shadowmancer is defeated, his shadowy form begins to crumble...                                                                               |
| The Shadowmancer     | thinking     | 8  | The protagonist outsmarts the Shadowmancer...                                                                                                     |
| The Shadowmancer     | tinkering    | 8  | The protagonist's action has effect on the Shadowmancer...                                                                                        |

Finally, the Companions and their characteristics are shown in Table 46. Similarly to before, it is possible to draw a correspondence with the natural language story outline and with the fields of the Companion class described in Section 3.2.

Table 46: Companions defined in the story

| Name               | Description                                                                                                                                                                                       | Ability Description                                                                                                              | Stats Bonuses                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|
| Golgi The Goblin   | A small, green goblin with a tuft of red hair on the top of his head and purple eyes. He carries explorer gears, and wears a worn-out leather armor.                                              | Golgi is a professional thief, with high stealth and sleight of hand ability. He is also able in combat with his old, dusty bow. | fighting: 1, athletics: 1, tinkering: 2 |
| Irina The Scholar  | A shy girl, with dark skin, black curly hair and black eyes. She wears a pair of big round glasses, a wizard hat and scholars robes.                                                              | Irina is a formidable scholar, very knowledgeable and smart, but is also a skilled mage with great spellcasting skills.          | thinking: 2, spellcasting: 2            |
| Miriam The Warrior | An extrovert, enthusiastic girl, with orange hair arranged in two braids, brown eyes and freckles on her cheeks. She wears a heavy plate armor with a red cloak and carries a two handed halberd. | Miriam is extremely skilled in combat, and is very strong and athletic.                                                          | fighting: 2, athletics: 2               |

### C.3. General considerations

By analyzing the two story outlines displayed before, the reader can appreciate how the largest amount of additional information added by human effort in the translation of the natural language outline to the formatted outline are not narrative information, rather parametric information (abilities, DCs, bonuses). This is probably the main challenge of the automatic generation of the formatted story outline, and the reason behind the consideration, expressed in Sections 3.4.2 and 7.1, that such an operation would require careful prompt engineering and more powerful or fine-tuned models. However, if the reader has any familiarity with LLMs, it would appear that this task is anything but impossible for modern language models.

Regarding the outline itself, as already expressed in Section 4.1, this outline produces stories too short to correctly evaluate the potential of the architecture. The issue is not only the limited number of rooms (only four in the whole story), but the linearity of the outline. The architecture and the application are projected to handle different paths inside the same narrative act, which might lead to different, interconnected locations (Rooms). Each Room might contain unique interactions, that can in turn change the course of other events by granting useful Items or Companions. This mechanism encourages and rewards exploration and curiosity by the user, other than increasing their freedom of action. In these scenarios, the architecture will structurally maintain and update the state of the adventure, making sure that it is consistent with respect to event location and room connections (net of errors of the underlying models, which might confuse or hallucinate elements), while the purely LLM solution will easily lose track of these complexities.

The limited branching and the small number of rooms (only four in total) constrained the scope of our evaluation. A more complex outline—with multiple interconnected rooms, conditional events, and meaningful dependencies—would have better demonstrated the architecture’s ability to manage state and maintain coherence. In the analyzed story outline, only one element employs, more or less, this mechanism: the `magical_key` obtained from Alken, which can help opening the Chest in the previous room. Nonetheless, even within these constraints, the architecture’s handling of this dependency (spanning two rooms and two events) offers a glimpse of its potential in more complex scenarios.

## Abstract in lingua italiana

Questa tesi presenta la progettazione, l'implementazione e la valutazione di un'architettura di scaffolding per la narrazione interattiva e immersiva, basata su *Large Language Models* (LLM) di piccole dimensioni (meno di 10 miliardi di parametri). L'architettura proposta supporta il processo di generazione attraverso componenti modulari per l'estrazione di informazioni, la generazione aumentata da recupero (RAG) e la gestione del contesto, con l'obiettivo di ridurre il carico computazionale e migliorare la coerenza a lungo termine in ambienti con risorse limitate.

L'architettura è stata implementata come sistema modulare basato sul modello Llama-3.1-8B-Instruct e testata rispetto ad un sistema puramente LLM in uno scenario di narrazione fantasy interattiva. È stato condotto uno studio utente su piccola scala, comprendente sette test con cinque utenti. Sono state effettuate analisi sia quantitative che qualitative, combinando questionari agli utenti (UEQ, SUS e valutazione personalizzata della storia) con metriche testuali automatiche relative a diversità lessicale, aderenza alla trama, aderenza all'input utente e coerenza semantica.

I risultati indicano che l'architettura di scaffolding non degrada significativamente la qualità percepita o oggettiva delle storie generate, offrendo al contempo notevoli vantaggi in termini di efficienza computazionale. In particolare, l'architettura mantiene sia i tempi di elaborazione che la lunghezza del contesto approssimativamente costanti al progredire della storia, mentre il sistema di riferimento mostra una crescita lineare in entrambe le dimensioni. Inoltre, vi sono evidenze che il sistema puramente LLM possa iniziare a degradarsi dopo circa 20 scene generate. Questi risultati supportano la conclusione che l'architettura proposta rappresenta una soluzione praticabile per la narrazione interattiva con piccoli LLM in contesti con risorse limitate, offrendo un bilanciamento favorevole tra qualità della storia e costo computazionale.

**Parole chiave:** architettura di scaffolding, narrazione interattiva, narrazione immersiva, modelli linguistici di grandi dimensioni, piccoli LLM, ambienti con risorse limitate, gestione del contesto, generazione narrativa.

## Acknowledgements

I would like to thank Prof. Franca Garzotto for the opportunity to work on this project, and also my co-advisors Giulia Valcamonica, Francesco Piferi, and Francesco Vona for their insights and support throughout this experience.

I want to express my sincere gratitude to the individuals who participated in the testing process. Their contributions offered important insights into my work and significantly enriched the value of this thesis.

I want to thank my parents, my sister, and all my family, who have never failed to encourage me and spur me on towards this goal.

Special thanks go to Silvia. You were my anchor and my north star, and you never missed an opportunity to cheer for me and make me feel better.

I also want to thank all my friends, far and near alike. In particular, those who introduced me to and helped me develop my passion for interactive storytelling and role-playing—the spark that made this work possible.

And to myself I say: thank you for how much you've grown. Do not even try to give up.