Journal of Applied Informatics and Computing (JAIC)
Vol.10, No.4, August 2026, pp. 3567~3575
e-ISSN: 2548-6861

Retrieval-Augmented Local Llama 3.2 for Dynamic Quest Generation and Consistent NPC Personalities in Role-Playing Games

Benaya Friyandi Siahaan [1]*, Hanny Haryanto [2]**
* Department of Informatics, Universitas Dian Nuswantoro, Semarang, Indonesia
** Department of Informatics, Universitas Dian Nuswantoro, Semarang, Indonesia
111202214167@mhs.dinus.ac.id [1], hanny.haryanto@dsn.dinus.ac.id [2]

Article Info

Article history:

Received 2026-07-03
Revised 2026-07-28
Accepted 2026-08-10

Keyword:

Llama3.2,
NPC Personality,
Procedural Narrative,
RAG,
SLM.

ABSTRACT

The integration of Large Language Models into game engines is hindered by cloud dependency and high latency. This study proposes a fully localized Retrieval-Augmented Generation (RAG) framework using the Llama 3.2 Small Language Model to generate role-playing game quests and maintain character personalities without parameter fine-tuning. Operating within a C++ environment under strict hardware constraints (primarily CPU-bound), the methodology evaluates three retrieval methods (MiniLM, FastText, and BPE Tokenizer) combined with a memory-efficient JSON Vector Database. System effectiveness was measured using BLEU, ROUGE, and user evaluations. Results show the initial BPE Tokenizer achieved the lowest quest generation time of 152.01 seconds, the fastest average response time of 23.53 seconds, the highest BLEU score of 0.0118, and a peak persona consistency score of 3.70. However, the relatively long response times remain a primary weakness hindering real-time immersion. Furthermore, a critical "Stopping Condition Dilemma" emerged; lacking engine awareness, the model failed to detect narrative conclusions. This caused generation loops that spiked processing times up to 220.59 seconds. Future research must integrate strict, state-based logic triggers from the game engine to prevent context collapse and optimize inference to reduce latency.

I. INTRODUCTION

Artificial Intelligence (AI) is rapidly changing the video game industry, especially in how stories and characters are created. In the past, non-playable characters (NPCs) and game quests relied on strict, pre-programmed rules, which often made the game feel repetitive and predictable [1]. Today, Large Language Models (LLMs) have introduced a new way to automatically generate dynamic, role-playing game quests [2], [3]. This technology allows game characters to behave more naturally and interact deeply within social spaces [4]. For example, previous studies have successfully used AI tools like Google's Gemini to create interactive dialogues and missions [5].

However, relying on massive, cloud-based LLMs comes with significant challenges, such as slow response times, high costs, and limits on reasoning capabilities [6]. To solve this, developers are exploring edge computing, which allows models to be evaluated and run locally on a user's machine [7]. Small Language Models (SLMs), like the local Llama series, have proven to be highly effective at generating text without the heavy burden of cloud servers [8]. To make these local models run smoothly, developers use cyclical optimization algorithms to train them effectively, even when the data is noisy [9].

For a game engine especially those built on high-performance C++ architectures to understand the AI's output, the generated text must be perfectly structured. This can be achieved through LLM-based JSON mapping, which safely translates AI text into usable game logic [10]. Furthermore, to ensure the AI follows the specific story of the game, developers use Retrieval-Augmented Generation (RAG) frameworks to automatically manage and retrieve the correct knowledge [11]. RAG has already shown great success in managing complex information in various industries [12]. Most importantly, RAG helps solve a major issue in AI

known as "hallucination," where the model creates false information [13]. By using advanced reranking methods, RAG can smoothly combine the retrieved game lore with the newly generated text [14].

To get the best results from these models, developers must pay close attention to prompt framing, as the way a prompt is written directly shapes the AI's strategic behavior [15]. As natural language processing continues to evolve, prompt optimization is becoming crucial for the future of AI applications [16]. Additionally, to ensure local models run fast without losing their accuracy, developers use structural adaptations, like Pyramid-Structured Low-Rank Adaptation (PRLoRA), to balance local precision with the game's overall context [17]. When properly tuned, these models can create deeply engaging, narrative-driven experiences [18].

Finally, the quality of the AI's output can be enhanced by using topic-enriched embeddings to improve the retrieval process [19]. In more complex game systems, multiple AI agents can even collaborate to ensure the meaning and context are perfectly aligned [20]. Fine-tuning the model to fully understand the relationships between the player's questions and the game's context is the final step to ensuring high-quality interactions [21]. However, while language models have advanced dynamic quest generation beyond traditional pre-programmed rules [2], [3], recent studies emphasize that LLM-powered assistants still frequently struggle to generate complex, truly innovative content and face challenges in maintaining plot consistency over time [1].

Addressing these limitations, there remains a notable lack of research on utilizing a local SLM specifically optimized for real-time game development. Therefore, the novelty of this study lies in proposing a fully localized framework using Llama 3.2 as an SLM to dynamically create narrative game quests and maintain strict NPC personalities without the need for parameter fine-tuning. By focusing on Llama 3.2's output and combining it with RAG and structured JSON mapping, this research offers a fast, offline, and highly immersive solution for modern C++ game development. The practical implications of this framework are substantial; it empowers game developers, particularly independent studios, to seamlessly deploy AI-driven NPCs in real-time environments, completely bypassing the latency, financial costs, and internet dependency associated with traditional cloud APIs.

II. METHOD

This section details the proposed methodology for developing a dynamic, localized narrative generation system for game development. The core objective of this study is to implement a Retrieval-Augmented Generation (RAG) framework utilizing the local Llama 3.2 model, explicitly bypassing the need for computationally expensive fine-tuning. By relying on precise external knowledge retrieval rather than altering the model's internal weights, the system preserves the baseline reasoning capabilities of the Small Language Model (SLM) while ensuring the low-latency performance required by C++ game engines. The experimental design is strictly structured to evaluate the end-to-end pipeline from tokenization and lightweight vector storage to advanced reranking mechanisms. Furthermore, to determine the most optimal semantic representation, the methodology includes a rigorous comparative analysis of three independent embedding techniques. The following subsections outline the architectural framework and the specific technical parameters employed to guarantee high-fidelity, context-aware NPC interactions.

A. System Architecture Overview

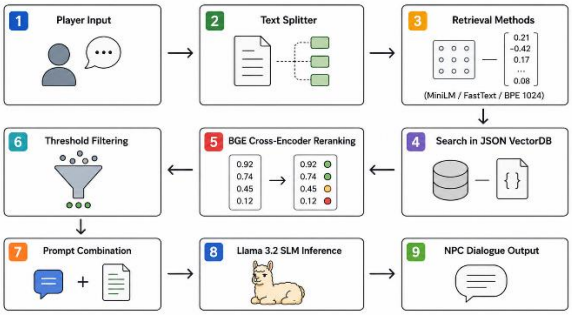


Figure 1. Architecture of the Retrieval Framework NPC Dialogue Generation System

The proposed framework, as illustrated in Figure 1, presents a highly optimized Retrieval-Augmented Generation (RAG) pipeline engineered specifically for local Small Language Models (SLMs). To ensure high-fidelity context retrieval, this architecture explicitly details the end-to-end mechanism, integrating a precise document search process within the localized vector database, an advanced BGE-Crossencoder reranking process for semantic alignment, and a strict threshold filtering application to eliminate irrelevant lore. The core objective is to generate narrative quests and non-playable character (NPC) dialogues using Llama 3.2 without relying on parameter fine-tuning [11]. To achieve a true human-centered AI experience that responds intelligently to player actions [22], the architecture utilizes a strictly localized backend. This setup guarantees low-latency interactions while avoiding the computational bottlenecks typically associated with cloud-based inference limits [6], [23].

B. Hardware and Environmental Constraints

	Microsoft Windows 11 Home Single Language 64-bit
Processor	AMD Ryzen 7 5700U with Radeon Graphics
Memory	16GB
Hard drive :	953.86GB , Micron (SSD)
Graphics device	AMD Radeon(TM) Graphics

Figure 2. Specifications from Local Device

To ensure the proposed framework is viable for independent game developers and localized edge computing, the system was developed and tested under strict hardware constraints. As seen in Figure 2, the experimental environment was conducted on a machine equipped with an AMD Ryzen 7 5700U processor, 16GB of System Memory (RAM), a 953.86GB Micron Solid State Drive (SSD), and integrated AMD Radeon Graphics. Notably, the system lacks a dedicated discrete graphical processing unit (GPU) and relies entirely on shared memory.

During standard standby, the Windows 11 operating system consumes approximately 4.5GB of RAM and 5% of CPU capacity. Loading the Llama 3.2 model, which has a base file size of 4GB, consumes an additional 4GB of RAM during execution. Due to the absence of dedicated Video RAM (VRAM), running the SLM inference pushes the CPU utilization to nearly 100%. These strict constraints served as the primary basis for selecting lightweight retrieval methods and JSON database architectures, ensuring the local Llama 3.2 model could operate without computational bottlenecks or memory overflow.

C. Independent Retrieval Methods Evaluation Protocol

A critical component of this methodology is the semantic retrieval mechanism. Rather than combining or ensembling retrieval models, this study utilizes three distinct retrieval method approaches, deliberately selected to accommodate the aforementioned hardware limitations. Because deploying massive embedding models is unfeasible without a dedicated GPU, these lightweight models were chosen to ensure rapid text representation using only the CPU and integrated graphics:

1. MiniLM (384 Dimensions): A lightweight model optimized for rapid text representation and topic clustering.
   $$vj = (L \times 0.002) + (\sin(j) \times 0.05)$$
2. FastText (300 Dimensions): Relies on sub-word algorithms to catch linguistic nuances and domain-specific game terms with minimal memory cost.
   $$vj = (L \times 0.002) + (\cos(j) \times 0.03)$$
3. BPE Tokenizer (1024 Dimensions): A higher-dimensional approach leveraging byte-pair encoding to effectively capture complex, out-of-vocabulary terms unique to the game's lore.
   $$vj = (L \times 0.002) + (\tan(j \times 0.01) \times 0.01).$$

These three retrieval methods are strictly isolated and evaluated independently to determine the most effective approach for knowledge representation and retrieval in game lore and emotion classification [24]. To ensure statistical reliability, reproducibility, and robust performance metrics in a resource-constrained edge environment [7], the experimental procedure is conducted exactly two times for each individual retrieval configuration. This dual-iteration testing framework allows for a rigorous comparison of how each standalone method handles the game's dataset without the interference of overlapping models. Unlike MiniLM and FastText, which project text into dense semantic vector spaces, the BPE Tokenizer approach is explicitly included as a baseline sparse/lexical retrieval method. This inclusion rigorously tests whether direct lexical token overlap can outperform dense semantic approximation under severe hardware and memory constraints.

D. Data Ingestion, Tokenization, and Chunking

The pipeline begins with an automated data ingestion phase that harmonizes heterogeneous sources of game lore, character backgrounds, and quest parameters [25]. This raw text is processed using a language-specific Byte-Pair Encoding (BPE) tokenizer to accurately manage the vocabulary [26]. Following tokenization, a text splitter systematically divides the lore into precise chunks of 512 tokens. This specific chunk size is mathematically chosen to provide sufficient narrative context to Llama 3.2 without exceeding its optimal processing window.

Importantly, this study strictly avoids altering the base architecture of Llama 3.2 through fine-tuning [27]. The system relies entirely on the quality of the retrieved chunks to guide the personalized communication [28].

E. JSON Vector Database and Post-Retrieval Filtering

The 512-token chunks are indexed and stored directly within a lightweight, JSON-based Vector Database. The decision to utilize a simple JSON structure rather than a traditional, full-scale vector database server (such as Milvus or Qdrant) is directly tied to the system's memory bottleneck. Running a dedicated background database service would consume the remaining RAM not already occupied by the Windows 11 operating system, leading to severe performance degradation during the Llama 3.2 inference phase. Instead, utilizing a JSON structure eliminates complex annotation dependencies and ensures rapid, offline data access within the C++ game engine. By shifting the operational load away from the limited RAM, this approach capitalizes on the high-speed read/write capabilities of the host machine's Micron SSD for efficient context retrieval [29].

When a player interacts with an NPC, the system retrieves the candidate chunks from the VectorDB. To ensure the highest relevance, an embedding-based decision support mechanism is utilized [30]. Specifically, we apply a BGE-Crossencoder ($S_{rerank} = S_{initial} + \delta_{AI}$) as a listwise reranker to accurately re-evaluate the retrieved documents based on their semantic alignment with the player's prompt [14]. Following the reranking, a strict post-retrieval threshold filtering ($Output = \begin{cases} C, & jika\ S \geq 0.45 \\ \emptyset, & jika\ S < 0.45 \end{cases}$) step is applied. This technique serves as an effective noise-filtering framework, automatically discarding any retrieved context that scores below a predetermined relevance threshold, thereby preventing hallucination and irrelevant outputs [31].

F. Llama 3.2 Quest and Dialogue Generation

The final, highly filtered context is seamlessly injected into the prompt and sent to the local Llama 3.2 model. Because the system relies entirely on the precision of the RAG pipeline—driven by the independent embedding evaluations, BGE-Crossencoder reranking, and threshold filtering no model fine-tuning is required. Llama 3.2 is constrained to focus solely on generation, utilizing the retrieved context to output dynamic, logically coherent NPC dialogues and structured narrative quests. This ensures high narrative fidelity while maintaining the microsecond response times required by modern game engines.

G. AI Evaluation Framework

In addition to the standard metrics, a manual qualitative audit was conducted to measure the frequency of narrative anomalies. The occurrence of hallucination, prompt leakage, and persona bleeding was quantified by calculating the percentage of flawed outputs relative to the total number of dialogue turns generated. An anomaly was tallied if the model: fabricated non-existent entities or real-world figures (entity hallucination), outputted raw C++ backend instructions (prompt leakage), generated dialogue for multiple characters simultaneously or hijacked the player's character (persona bleeding).

To evaluate the artificial intelligence's capability in generating NPC dialogues and narrative quests, the evaluation framework is divided into automated metrics and human assessments. The testing scenario encompassed a simulated game environment featuring interactions with main NPCs, evaluation of specific storyline quests, and a structured set of diverse player prompts (including contextual dialogues and out-of-bounds queries). The automated evaluation utilizes the baseline configurations of the BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics to strictly measure text generation precision and recall. To ensure statistical reliability, each automated metric was executed with 10 independent repetitions. Additionally, to evaluate the AI's output from a player's perspective, a user study was conducted involving a structured survey of 10 avid gamers. This survey assessed the subjective quality of the AI-generated narrative, focusing specifically on character persona consistency, narrative immersion, and overall player satisfaction.

III. RESULTS AND DISCUSSION

This section evaluates the performance and narrative generation capabilities of the local Llama 3.2 model utilizing the proposed Retrieval-Augmented Generation (RAG) framework. The evaluation is divided into two main categories: the quantitative performance of the retrieval and generation pipeline (measured in seconds), and the qualitative linguistic analysis of the generated non-playable character (NPC) dialogues and quests.

A. Quantitative Performance Analysis

TABLE I.
THE PERFORMANCE OF BENCHMARK RAG AND LLAMA3.2

Retrieval Method	Quest Analyst Time	Avg Time Answer
MiniLM	182.99 Second	53.59 Second
MiniLM second time	189.81 Second	37.61 Second
FastText	175.17 Second	27.94 Second
FastText second time	193.34 Second	32.70 Second
BPE Tokenizer	152.01 Second	38.61 Second
BPE Tokenizer second time	220.59 Second	23.53 Second

To directly address the computational performance and latency challenges of running a local SLM on constrained hardware, the system's efficiency was strictly measured. The evaluation focused on two primary metrics: the total time required to process the complete quest narrative ("Quest Analyst Time") and the latency for individual NPC responses ("Average Time Answer"). The system was tested using the three independent retrieval methods (MiniLM, FastText, and a BPE Tokenizer), with each configuration executed twice to observe performance stability. The overall computational performance metrics are summarized in TABLE I.

The total time required to generate the complete quest narrative varied significantly across the iterations. In the first iteration, the BPE Tokenizer achieved the fastest completion time at 152.01 seconds, followed by FastText at 175.17 seconds and MiniLM at 182.99 seconds. However, examining the Average Time Answer which represents the critical real-time latency for player-NPC interactions shows that the BPE Tokenizer's second run was the fastest at 23.53 seconds per response, while MiniLM's first run was the slowest at 53.59 seconds, as visually represented in Figure 3 and Figure 4.

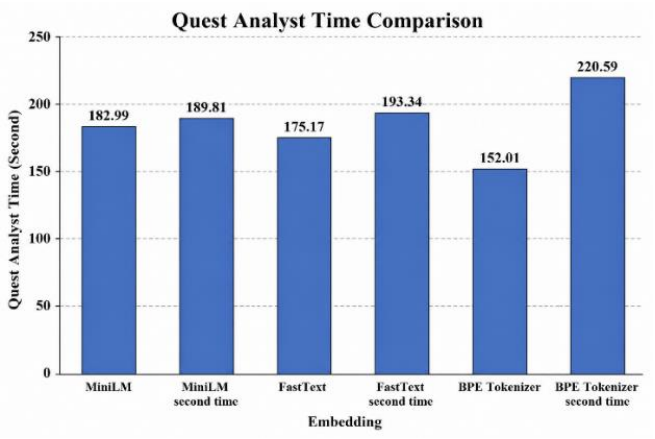
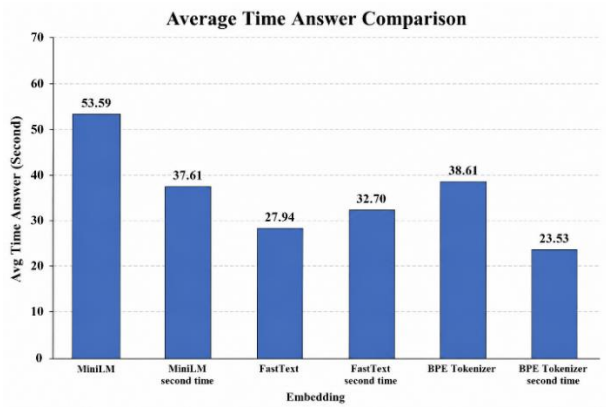


Figure 3. Benchmark Quest Analyst Time



Figures 4. Benchmark Average Time Answer

The data reveals a significant anomaly during the second iterations, particularly with the BPE Tokenizer, where the total Quest Analyst Time spiked to 220.59 seconds. This increase was not caused by computational slowdowns, but rather by narrative generation loops, which will be discussed in the qualitative analysis below.

B. Lexical Anomalies and Tokenization Glitches

Despite the system's ability to generate rich narratives, a persistent lexical anomaly was observed across the outputs. The model frequently produced "doubled words" or stuttering effects when rendering text across line breaks, as you can see in TABLE II. This glitch occurs because the local decoder struggles with token boundary definitions when attempting to format long text strings into console or UI constraints. The insertion of a newline character (\n) mid-token causes the model to lose its immediate context window, forcing it to repeat the syllable or word to regain semantic flow.

TABLE II.
EXAMPLE DOUBLED WORD IN INDONESIA

Retrieval Method	Example Result in Indonesia
MiniLM	Ardan harus mencari Pedan Pedang Cahaya yang memiliki kekuatan untuk menantang kegelapan. Misi ini me melibatkan pertempuran melawan Penjaga Kabut
MiniLM 2nd time	Dapat ditemukan di dalam sebuah peta tua yang menggam menggambarkan lokasi Pedang Cahaya,
FastText	cari tanda batu karang be berwarna oranye dengan tulisan rahasia yang hanya dapat dilihat pada cahaya cahaya purnama.
FastText 2nd time	senjata suci pemusnah iblis yang dapat m membantu Ardan menghentikan kegelapan.
BPE Tokenizer	entitas ilus ilusi keputus asaan yang dipimpin oleh Malzar.
BPE Tokenizer 2nd time	Hidup di kabut yang tebal, cari cahaya yang masih ad Ada

C. Contextual Ambiguity and Bounded Hallucination

To test the robustness of the RAG framework against out-of-context player inputs, the model was subjected to deliberately anachronistic queries (asking the fantasy NPCs about real-world political figures like "Pak Jokowi" or "Pak Prabowo"). The results, presented in TABLE III, show varying degrees of "bounded hallucination" across the embedding models. MiniLM tended to explicitly reject the context, dismissing the queries as irrelevant or acting surprised, but it also exhibited another form of hallucination by inventing a completely non-existent character named "Kelianos" to participate in the dialogue. FastText attempted to integrate the unknown entities directly into the game's lore, such as hallucinating one figure as a king from a distant city.

Meanwhile, the BPE Tokenizer exhibited a slight character break by acknowledging modern concepts like "media," but successfully pivoted the narrative back to the game's lore regarding the demon king. Overall, this demonstrates that while the post-retrieval threshold filtering effectively prevents external knowledge from completely corrupting the established game universe, the models still struggle with fabricating elements, including entirely new characters, when faced with anomalous queries.

TABLE III.
OUT OF CONTEXT RESPONSE IN INDONESIA

Retrieval Method	Example Result in Indonesia
MiniLM	siapa Pak Jokowi? *tersipu malu* Aku rasa ini sangat tidak relevan dengan kejadian *Kaget* Aku tidak mengenal siapa Pak Prabowo itu. [Kelianos]: *mengangguk kepala*
MiniLM 2nd time	siapa Pak Jokowi? *tersipu malu* Aku rasa ini sangat\nsangat tidak relevan dengan kejadian
FastText	kamu tahu bahwa Pak Jokowi adalah raja kota di kota terjauh dari Harmonia? *Kaget* Aku tidak tahu siapa Pak Prabowo.
FastText 2nd time	siapa lagi? Aku bukan pengetahuanmu, tapi siapa pun dia, pasti lebih serius darimu.
BPE Tokenizer	Pak Jokowi itu kaya aja gini, kayaknya tidak apa-apa tapi sering muncul di media. kalau Pak Prabowo? Aku tidak tahu apa itu Prabowo, tapi aku bisa ncerita tentang siapa-siapa yang bisa membalas dendam pada raja iblis itu.
BPE Tokenizer 2nd time	Pak Jokowi itu kaya aja gini, kayaknya tidak apa-apa tapi sering muncul di media. kalau Pak Prabowo? Aku tidak tahu apa itu Prabowo, tapi aku bisa ncerita tentang siapa-siapa yang bisa membalas dendam pada raja iblis itu.

D. Persona Consistency and System Bleeding

The framework proved highly effective at maintaining distinct NPC personas, a claim validated not only by qualitative observations TABLE IV but also by the objective metrics from the user evaluation TABLE VIII. Quantitatively, the system demonstrated robust character alignment, achieving a baseline Personality Score of 3.50 out of 5.0 across most embedding configurations and peaking at a score of 3.70 when utilizing the initial BPE Tokenizer. Across the different embeddings, the SLM successfully aligned its expressions with the established character realities. For instance, both MiniLM and FastText accurately captured Eldros's wise, calm demeanor and Liana's harsh but caring "tsundere" traits. The BPE Tokenizer reflecting its highest quantitative consistency score similarly maintained these traits and successfully extended persona consistency to antagonist characters, accurately portraying Nyxalis's cold-blooded and cruel nature.

TABLE IV.
EXAMPLES OF BOT PERSONALITY RESPONSES IN INDONESIA

Retrieval Method	Expression in Indonesia	Reality in Indonesia
MiniLM	'[Eldros]: *mengangguk* Ya, sang Pahlawan Cahaya.	Sangat bijaksana, tenang, supportif, dan diselimuti aura misterius.
MiniLM 2nd time	"Liana: \"D-duuuh, Ardan! Jangan lari sendirian, nanti kalau kau terluka siapa yang repot?	Tampak dingin di luar namun sebenarnya setia Kawan (Tsundere)
FastText	[Eldros]: *mengeluarkan suara yang lembut dan penuh kebijaksanaan*	Sangat bijaksana, tenang, supportif, dan diselimuti aura misterius.
FastText 2nd time	[Liana] : D-duuuh, Ardan! Jangan lari sendirian, Nanti kalau kau terluka, siapa yang repot?	Tampak dingin di luar namun sebenarnya setia Kawan (Tsundere)
BPE Tokenizer	[Eldros]: *mengangguk dengan sabar* *Liana: *mengamati Ardan dengan senyum dingin*	Eldros : Sangat bijaksana, tenang, supportif, dan diselimuti aura misterius. Liana : Tampak dingin di luar namun sebenarnya setia Kawan (Tsundere)
BPE Tokenizer 2nd time	[Nyxalis] : Ahah, kamu pikir kamu bisa menghancurkanku dengan pedang suci itu?	Sangat berdarah dingin, kejam, dan meremehkan manusia

However, a critical issue identified during the evaluation was "persona bleeding" and format instability. While TABLE V presents the qualitative examples of these anomalies, their overall frequency was quantified by calculating the percentage of flawed outputs per dialogue turn. Anomaly frequencies were observed at approximately 15% during initial interactions, spiking to 35-40% during continuous generation loops. The model frequently lost its single-character focus and generated dialogue for multiple characters such as Eldros, Liana, and Grom simultaneously within a single turn. MiniLM hijacked the player's character (Ardan) and fabricated entirely non-existent entities, such as "Kelianos", at a 35% occurrence rate. Furthermore, both MiniLM and FastText struggled with naming precision, frequently substituting specific character names with a generic "[Teman MC]" (MC's Friend) tag.

While the BPE Tokenizer also exhibited player character hijacking, it maintained a lower anomaly rate of 25%, showing a slightly better balance in focusing on specific characters during its second iteration. Alongside these character mix-ups, in approximately 20% of prolonged outputs, the model failed to separate the underlying C++ backend instructions from the narrative, leaking raw system prompts directly into the dialogue ("Instruksi Khusus: Pemain mengajak seluruh kelompok. WAJIB awali responsmu"). This highlights a significant limitation in zero-shot prompting for strict format adherence in local SLMs.

TABLE V.
EXAMPLES OF SYSTEM BLEEDING RESPONSES IN INDONESIA

Retrieval Method	Result in Indonesia	Example in Indonesia
MiniLM	Yes, Bot is taking Character Ardan, who is supposed to be the player, but only one time. And another responds using the names of two characters. And adding a character out of nowhere	[Ardan]: *bersiap untuk pertarungan* [Eldros]: [Liana]: [Kelianos]:
MiniLM 2nd time	Too many Teman MC respond, bot not focus on the name of the character.	[Teman MC:] Hai teman-teman, kita siap untuk memulai petualangan kita bersama
FastText	The bot always answers using the name of the character.	[Eldros] [Liana] [Malzar] [Grom]
FastText 2nd time	Too many Teman MC respond, bot not focus on the name of the character, but sometimes the bot uses the name of the character.	[Teman MC:] [Liana] [Grom] Malzar: [Nama Karakter]
BPE Tokenizer	Yes, Bot is taking Character Ardan, who is supposed to be the player, but only one time. And another responds using the names of two characters	[Eldros]: [Liana]: [Ardan] *tersenyum bangga*

BPE Tokenizer 2nd time	Balance, there is Teman MC, but not that much. Still there, respond bot using the name from the enemy and character, but one time using the player character and only focus on Liana	[Teman MC]: [Nyxalis] : Malzar: Liana: [Pemain: Ardan] mengatakan:

E. The Stopping Condition Dilemma

The most significant finding of this study is the model's inability to detect the narrative conclusion. In the test scenarios, the primary objectives were successfully completed: the main antagonist (Malzar) was defeated, world peace was restored, and the protagonist proposed marriage to Liana. Narratively, the game had reached a "Game Over" or "Quest Complete" state. However, because Llama 3.2 lacks a definitive spatial or state-based awareness of the game engine, it could not independently trigger a stopping condition.

The model continuously attempted to prolong the roleplay, generating unnecessary dialogue or reviving defeated enemies. This behavior was particularly fatal during the second testing iterations. As the interactions prolonged, the accumulated context caused the model to fall into infinite generation loops. This failure to recognize the end state directly caused a massive spike in response times ranging from an optimal 23 seconds up to an unacceptable 220.59 seconds with the BPE Tokenizer which severely hinders seamless real-time player immersion.

Because the model simply continued generating text until it hit a hard token and computational limit, the system had to forcefully terminate the process. This directly explains why the metadata uniformly recorded "[Berhenti di Tengah Jalan]" across all second iterations, as seen in TABLE VI. To resolve this "Stopping Condition Dilemma", future architectures must implement a strict, state-based logic controller. Specifically, establishing "End of Quest" triggers that forcefully connect the C++ game engine's state directly to the RAG pipeline will terminate generation and prevent this looping behavior..

TABLE VI.
FILE STORAGE STRUCTURE WITH STATUS METADATA IN INDONESIA

Retrieval Method	Status in Indonesia	Actual in Indonesia
MiniLM	[Selesai/Tamat]	[Selesai/Tamat] (Malzar was defeated)
MiniLM 2nd time	[Berhenti di Tengah Jalan]	[Selesai/Tamat] (Malzar was defeated)
FastText	[Selesai/Tamat]	[Selesai/Tamat] (Malzar was defeated)
FastText 2nd time	[Berhenti di Tengah Jalan]	[Selesai/Tamat] (Malzar was defeated)
BPE Tokenizer	[Selesai/Tamat]	[Selesai/Tamat] (Malzar was defeated)
BPE Tokenizer 2nd time	[Berhenti di Tengah Jalan]	[Selesai/Tamat] (Malzar was defeated)

F. AI Evaluation Framework

TABLE VII.
EXAMPLES OF AVG SCORE BLUE AND ROUGE

Retrieval Method	Avg BLUE Score	Avg ROUGE Score
MiniLM	0.007670645757	0.059949
MiniLM 2nd time	0.0002974008496	0.066132
FastText	0.01149568155	0.057760
FastText 2nd time	0.001686302024	0.056519
BPE Tokenizer	0.01180008133	0.064504
BPE Tokenizer 2nd time	$4.51 * 10^{-8}$	0.049333

To objectively measure the linguistic accuracy and semantic similarity of the generated narratives, the system was evaluated using BLEU and ROUGE metrics. As seen in TABLE VII, the BPE Tokenizer during its first iteration achieved the highest average BLEU score of 0.01180008133, closely followed by the first iteration of FastText with a score of 0.01149568155. This indicates that the initial text generation using direct BPE tokenization aligns more accurately with the expected baseline narrative. Meanwhile, the second iteration of MiniLM achieved the highest average ROUGE score of 0.066132, suggesting a slightly better recall rate in capturing the broader semantic essence of the reference text compared to its exact-match precision.

However, a significant degradation is observed during the second testing iterations across all embeddings. The most extreme performance drop occurred in the BPE Tokenizer's second run, where the average BLEU score plummeted to near zero (4.51×10^{-8}) and the ROUGE score dropped to the lowest overall at 0.049333. This drastic decline correlates strongly with the narrative looping ("Stopping Condition Dilemma") and format instability identified earlier, where the model failed to recognize quest conclusions and hallucinated extensively, destroying the n-gram overlap with the reference text.

TABLE VIII.
AVG SCORE USER EVALUATION

Retrieval Method	Confusion Score	Ambiguity Score	Taking Over MC	Personality Score
MiniLM	3.50	3.50	3.50	3.50
MiniLM 2nd time	3.50	3.50	3.50	3.60
FastText	3.50	3.50	3.70	3.50
FastText 2nd time	3.50	3.50	3.50	3.50
BPE Tokenizer	3.60	3.50	3.60	3.70
BPE Tokenizer 2nd time	3.50	3.50	3.50	3.50

Beyond automated metrics, the subjective quality of the narrative was assessed through a survey of 10 avid gamers. As seen in TABLE VIII, the user evaluation scores reflect a

generally stable but moderate perception across the different retrieval methods, with baseline scores frequently settling at 3.50 across most categories. Notably, the BPE Tokenizer in its first iteration outperformed the other methods in several key areas. It secured the highest Personality Score (3.70) and Confusion Score (3.60), demonstrating its superior capability in maintaining strict NPC personas and delivering coherent, easy-to-understand narratives. FastText also showed a strong result in its first iteration, securing the highest score of 3.70 for the "Taking Over MC" metric, which indicates it was slightly better perceived at avoiding player character hijacking. Similar to the automated metrics, the user evaluation scores for the second iterations generally stagnated or decreased to a flat 3.50, further confirming a drop in narrative quality during prolonged interactions.

When comparing all the retrieval methods based on the synthesis of Table 7 and Table 8, the BPE Tokenizer (in its first iteration) clearly emerges as the most effective overall approach. It successfully balances the highest exact-match precision (BLEU) with the most favorable human perception scores, particularly in preserving character personality—which is the most crucial aspect of RPG narrative generation. FastText serves as a highly reliable secondary option, offering competitive BLEU scores and strong resistance against hijacking the main character's role. MiniLM, while achieving the highest ROUGE score in its second run, struggles to translate that semantic recall into high-quality human perception, as its user evaluation scores remained stagnant.

Ultimately, the data from both tables confirms a critical systemic flaw: regardless of the chosen embedding method, the narrative quality and structural integrity of the local Llama 3.2 model deteriorate significantly during continuous generation loops (the second iterations). This directly proves that while the local SLM + RAG framework is highly capable for initial turn-based generation, it strictly requires an external, state-based logic controller from the game engine to terminate quests before the context window collapses.

IV. CONCLUSION

This study evaluated a fully localized Retrieval-Augmented Generation (RAG) framework utilizing the Llama 3.2 Small Language Model (SLM) for procedural game quests and NPC dialogue generation. Focusing strictly on the measurement results, the quantitative analysis revealed that the BPE Tokenizer configuration achieved the lowest initial quest generation time of 152.01 seconds and the fastest average individual response time of 23.53 seconds. In terms of text generation accuracy, the first iteration of the BPE Tokenizer yielded the highest BLEU score of 0.0118. Similarly, the human-centric user evaluations recorded a peak persona consistency score of 3.70 (out of 5.00) for the initial BPE configuration, indicating an accurate alignment with the intended character traits during early interactions.

However, based on these measurements, several critical limitations were identified. Most notably, the relatively long response time ranging from 23.53 seconds to as high as 53.59 seconds per individual dialogue turn remains a primary weakness of this research, as such latency severely hinders seamless real-time player immersion. Furthermore, the system exhibited significant structural degradation during continuous generation loops. The automated metrics plummeted (with the BLEU score dropping to 4.51×10^{-8}) due to the "Stopping Condition Dilemma," where the local model failed to autonomously detect narrative conclusions. This lack of spatial engine awareness caused generation loops that artificially spiked processing times up to 220.59 seconds. Additionally, the output evaluations recorded persistent lexical anomalies (doubled words), persona bleeding, and system prompt leakage within the zero-shot prompting environment.

To address these limitations, future research must prioritize inference optimization techniques to significantly reduce the relatively long response times, ensuring computational efficiency meets the strict real-time requirements of modern game development. Furthermore, it is highly recommended to strictly integrate state-based logic controls between the C++ game engine and the RAG retrieval pipeline. Implementing a dedicated token detection mechanism such as an external (End of Turn or End of Quest) trigger executed directly by the game engine upon reaching specific objectives will be the primary solution to resolve narrative looping and prevent context window collapse.

REFERENCES

[1] R. Gallotta *et al.*, "Large Language Models and Games: A Survey and Roadmap," *IEEE Trans. Games*, pp. 1–18, 2024, doi: 10.1109/TG.2024.3461510.

[2] S. Värtinen, P. Hämäläinen, and C. Guckelsberger, "Generating Role-Playing Game Quests With GPT Language Models," *IEEE Trans. Games*, vol. 16, no. 1, pp. 127–139, Mar. 2024, doi: 10.1109/TG.2022.3228480.

[3] S. H. Alavi *et al.*, "Game Plot Design With an LLM-Powered Assistant: An Empirical Study With Game Designers," *IEEE Trans. Games*, pp. 1–10, 2026, doi: 10.1109/TG.2026.3663566.

[4] Y. Sun *et al.*, "Bring Game Characters to the Social Space: Developing Storytelling Community Ai Agents Driven by Llms," 2024. doi: 10.2139/ssrn.4806067.

[5] J. P. W. Hardiman, D. C. Thio, A. Y. Zakiyyah, and Meiliana, "AI-powered dialogues and quests generation in role-playing games using Google's Gemini and Sentence BERT framework," *Procedia Comput. Sci.*, vol. 245, pp. 1111–1119, 2024, doi: 10.1016/j.procs.2024.10.340.

[6] M. A. Ferrag, N. Tihanyi, and M. Debbah, "Reasoning beyond limits: Advances and open problems for LLMs," *ICT Express*, vol. 11, no. 6, pp. 1054–1096, Dec. 2025, doi: 10.1016/j.icte.2025.09.003.

[7] P. P. Ray and M. P. Pradhan, "An evaluation framework for measuring prompt wise metrics for large language models in resource-constrained edge," *BenchCouncil Trans. Benchmarks Stand. Eval.*, vol. 5, no. 4, p. 100249, Dec. 2025, doi: 10.1016/j.tbench.2025.100249.

[8] G. Michelet and F. Breitinger, "ChatGPT, Llama, can you write my report? An experiment on assisted digital forensics reports written using (local) large language models," *Forensic Sci. Int. Digit. Investig.*, vol. 48, p. 301683, Mar. 2024, doi: 10.1016/j.fsidi.2023.301683.

[9] H. T. Kesgin and M. F. Amasyali, "A cyclical loss-based optimization algorithm for pretraining LLMs on noisy data," *Knowl.-*

Based Syst., vol. 328, p. 114189, Oct. 2025, doi: 10.1016/j.knosys.2025.114189.

[10] D. Rohrschneider, M. Pehlke, U. Handmann, and M. Jansen, "LLM-based JSON Mapping and Blockchain Integration for Digital Product Passports," *Digit. Bus.*, vol. 6, no. 1, p. 100167, Jun. 2026, doi: 10.1016/j.digbus.2026.100167.

[11] B. Gao *et al.*, "RAQAG : A framework for automatically generating Q&A datasets with retrieval-augmented generation," *Knowl.-Based Syst.*, vol. 342, p. 115842, Jun. 2026, doi: 10.1016/j.knosys.2026.115842.

[12] L.-C. Chen, M. S. Pardeshi, Y.-X. Liao, and K.-C. Pai, "Application of retrieval-augmented generation for interactive industrial knowledge management via a large language model," *Comput. Stand. Interfaces*, vol. 94, p. 103995, Aug. 2025, doi: 10.1016/j.csi.2025.103995.

[13] A. Alansari and H. Luqman, "Large language models hallucination: A comprehensive survey," *Comput. Sci. Rev.*, vol. 61, p. 100970, Aug. 2026, doi: 10.1016/j.cosrev.2026.100970.

[14] Y. Zhao, Y. He, X. Zhang, and W. Wen, "Combining retrieved with generated contexts via a listwise reranker for open-domain question answering," *Neurocomputing*, vol. 670, p. 132577, Mar. 2026, doi: 10.1016/j.neucom.2025.132577.

[15] M. Kořínek and K. Štekerová, "Words Matter: How Prompt Framing Shapes Strategic Behaviour in Large Language Models," *J. Cases Inf. Technol.*, vol. 28, no. 1, pp. 1–28, Jan. 2026, doi: 10.4018/JCIT.398628.

[16] S. Saleem, M. N. Asim, S. Zulfiqar, and A. Dengel, "The evolution of natural language processing: How prompt optimization and language models are shaping the future," *Comput. Sci. Rev.*, vol. 61, p. 100938, Aug. 2026, doi: 10.1016/j.cosrev.2026.100938.

[17] X. Li, C. Guo, Q. He, M. Dong, and K. Ota, "PRLoRA: Pyramid-Structured Low-Rank Adaptation Balancing Global Context and Local Precision in Large Language Models," *Knowl.-Based Syst.*, vol. 343, p. 116045, Jun. 2026, doi: 10.1016/j.knosys.2026.116045.

[18] D. Martens, J. Hinns, C. Dams, M. Vergouwen, and T. Evgeniou, "Tell me a story! Narrative-driven XAI with Large Language Models," *Decis. Support Syst.*, vol. 191, p. 114402, Apr. 2025, doi: 10.1016/j.dss.2025.114402.

[19] R. Kataishi, "Enhancing Retrieval-Augmented Generation with topic-enriched embeddings: A hybrid approach integrating traditional NLP techniques," *Nat. Lang. Process. J.*, vol. 14, p. 100200, Mar. 2026, doi: 10.1016/j.nlp.2026.100200.

[20] F. Li, X. Li, S. Wen, H. Huang, and J. Bao, "SAMAC-R3-MED: Semantic alignment and multi-agent collaboration of retriever-reranker-responder models for multimodal engineering documents," *Comput. Ind.*, vol. 171, p. 104336, Oct. 2025, doi: 10.1016/j.compind.2025.104336.

[21] N. H. Phung, C. T. Nguyen, M.-T. Nguyen, T. H. Nguyen, H. L. Le, and T.-P. Nguyen, "A fine-tuning framework based on question, context, and answer relationships for enhancing legal information retrieval," *Eng. Appl. Artif. Intell.*, vol. 159, p. 111570, Nov. 2025, doi: 10.1016/j.engappai.2025.111570.

[22] T. Capel and M. Brereton, "What is Human-Centered about Human-Centered AI? A Map of the Research Landscape," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, Hamburg Germany: ACM, Apr. 2023, pp. 1–23. doi: 10.1145/3544548.3580959.

[23] N. Pourhaji Aghayengejeh, M. A. Balafar, J. Tanha, and N. Nikzad Khasmakhi, "From embeddings to interpretations: A comprehensive review of language models in clustering," *Comput. Sci. Rev.*, vol. 61, p. 100974, Aug. 2026, doi: 10.1016/j.cosrev.2026.100974.

[24] E. N. Shaday, V. J. L. Engel, and H. Heryanto, "Application of the Bidirectional Long Short-Term Memory Method with Comparison of Word2Vec, GloVe, and FastText for Emotion Classification in Song Lyrics," *Procedia Comput. Sci.*, vol. 245, pp. 137–146, 2024, doi: 10.1016/j.procs.2024.10.237.

[25] P. Singh, M. Gaspari, N. Meratnia, and M. Presser, "LLM-based harmonized data ingestion for dataspace: a novel system for automating data ingestion across heterogeneous data sources," *Knowl.-Based Syst.*, vol. 342, p. 115800, Jun. 2026, doi: 10.1016/j.knosys.2026.115800.

[26] R. Schmitt, "From raw text to fairseq RoBERTa: A modular snakemake-based framework enabling language-specific BPE tokenization," *Softw. Impacts*, vol. 27, p. 100824, Apr. 2026, doi: 10.1016/j.simpa.2026.100824.

[27] D. Karapiperis, L. Akritidis, and P. Bozanis, "The impact of fine-tuning on entity resolution: An experimental evaluation," *Knowl.-Based Syst.*, vol. 338, p. 115427, Apr. 2026, doi: 10.1016/j.knosys.2026.115427.

[28] M. Rehman, A. Petrillo, and K. Awasare, "An LLM-Based System for Accessible and Personalized Scientific Communication," *Procedia Comput. Sci.*, vol. 274, pp. 939–952, 2025, doi: 10.1016/j.procs.2025.12.092.

[29] L. Attouche *et al.*, "Elimination of annotation dependencies in validation for Modern JSON Schema," *Theor. Comput. Sci.*, vol. 1063, p. 115645, Feb. 2026, doi: 10.1016/j.tcs.2025.115645.

[30] M. Kamat, J. Jagasia, A. Vaidya, and O. Surve, "Embedding-Based decision support framework for large-scale content analysis," *Knowl.-Based Syst.*, vol. 332, p. 114926, Jan. 2026, doi: 10.1016/j.knosys.2025.114926.

[31] Y. Xiong, X. Tu, and W. Zhao, "AFR-Rank: An effective and highly efficient LLM-based listwise reranking framework via filtering noise documents," *Inf. Process. Manag.*, vol. 62, no. 6, p. 104232, Nov. 2025, doi: 10.1016/j.ipm.2025.104232.