Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Stellar Rune Powered By DGR AI – A Deep Learning-Powered Conversational Intelligence For Immersive 2D Gaming

Siddhanth Muttoo[1], Dipti Jadhav[2*] Sumedha Bhagwat[3], Rajashree Shedge[4], Tushar Ghorpade[5], Swapnil Shinde[6]

[1]Department of Computer Science and Engineering, Ramrao Adik Institute of Technology, D Y Patil Deemed to be University, Nerul, Navi Mumbai, Maharashtra, India. Email Id: siddhanthmuttoo@gmail.com
[2*]Department of Computer Science and Engineering, Ramrao Adik Institute of Technology, D Y Patil Deemed to be University, Nerul, Navi Mumbai, Maharashtra, India. Email Id: dipti.jadhav@rait.ac.in
[3]Department of Information Technology, Ramrao Adik Institute of Technology, D Y Patil Deemed to be University, Nerul, Navi Mumbai, Maharashtra, India Email Id sumedha.bhagwat@rait.ac.in
[4]Department of Computer Science and Engineering, Ramrao Adik Institute of Technology, D Y Patil Deemed to be University, Nerul, Navi Mumbai, Maharashtra, India. Email Id: rajashree.shedge@rait.ac.in
[5]Department of Computer Engineering, Ramrao Adik Institute of Technology, D Y Patil Deemed to be University, Nerul, Navi Mumbai, Maharashtra, India. Email Id: tushar.ghorpade@rait.ac.in
[6]Department of Engineering and Technology, Bharati Vidyapeeth Deemed to be University, Kharghar, Navi Mumbai, Maharashtra, India Email id: swapnil.shinde1@bharatividyapeeth.edu

ABSTRACT
This paper presents Stellar Rune, a 2D conversational intelligence game designed to leverage deep learning techniques for an improved immersive player experience. At the core of Stellar Rune is the DGR-AI engine: a novel deep learning-based intent classification system that integrates both neural network architectures and semantic embeddings for robust natural language understanding. The proposed model leverages Sentence-BERT embeddings, which transforms the user queries into dense vector representations that preserve semantic meaning. These embeddings are then processed through a stacked architecture comprising Conv1D, Bidirectional LSTM (BiLSTM), and Dense layers, enabling the system to capture both local n-gram features and long-range contextual dependencies. A tokenization and embedding pipeline ensure efficient input pre-processing, while the neural classifier refines predictions across multiple intent classes. To further enhance reliability, the proposed system incorporates an Intent Override mechanism and a Fallback model. The override ensures correction of ambiguous or misclassified intents, while the fallback addresses out-of-scope queries using alternative matching strategies. The Stellar Rune framework is evaluated based on cross-entropy loss, intent accuracy, predicted accuracy, and actual accuracy metrics, combining correct neural predictions, semantic matches, and fallback resolutions. The proposed architecture demonstrates improved performance by unifying neural predictions with semantic similarity checks, offering a scalable solution for intent detection, natural language processing (NLP), and conversational AI. This hybrid framework, showcased through the Stellar Rune platform, highlights the effectiveness of combining transformer-based embeddings with deep learning classifiers for real-world dialog systems.

Keywords: Conversational-AI, 2d-gaming, GPT, Transformer, Non-playable characters, BERT, Sentence- BERT, intents, Deep Learning, Artificial Intelligence.

INTRODUCTION

The field of game development has consistently sought new methods to deepen player engagement, and one of the most underutilized areas of innovation lies in character interaction—specifically, how non-playable characters (NPCs) communicate with the players. Modern graphics and gameplay systems have made significant improvements, but most dialogue systems remain linear, repetitive, or static. The proposed project, Stellar Rune, addresses this gap by embedding a conversational artificial intelligence system, DGR AI, directly into a 2D game environment. Unlike typical chatbot overlays or scripted lines, Stellar Rune uses DGR AI as an in-game component that brings a new level of immersion through intelligent language interaction.

The uniqueness of Stellar Rune is in its fusion of natural language understanding and 2D world simulation, offering a blend of atmospheric world design and intelligent responsiveness powered by DGR AI. It is set

in a pixel-art style map built using Tiled Map Editor and integrated with an interactable environment using the Pygame framework. Stellar Rune visually conveys the player's mood and presence through dynamic elements like day-night cycles, animated weather effects,

and environmental lighting. The player can navigate a rich map that includes forests, houses, caves, and lakes—each inhabited by uniquely placed NPCs. In this system, the simulated world is not just a static background; it is shown as a canvas for personalized, context aware, intent driven dialogue that change depending on what the player commands through the DGR AI model. The screenshot of the working model of Stellar Rune is as shown in Figure 1 and 2.

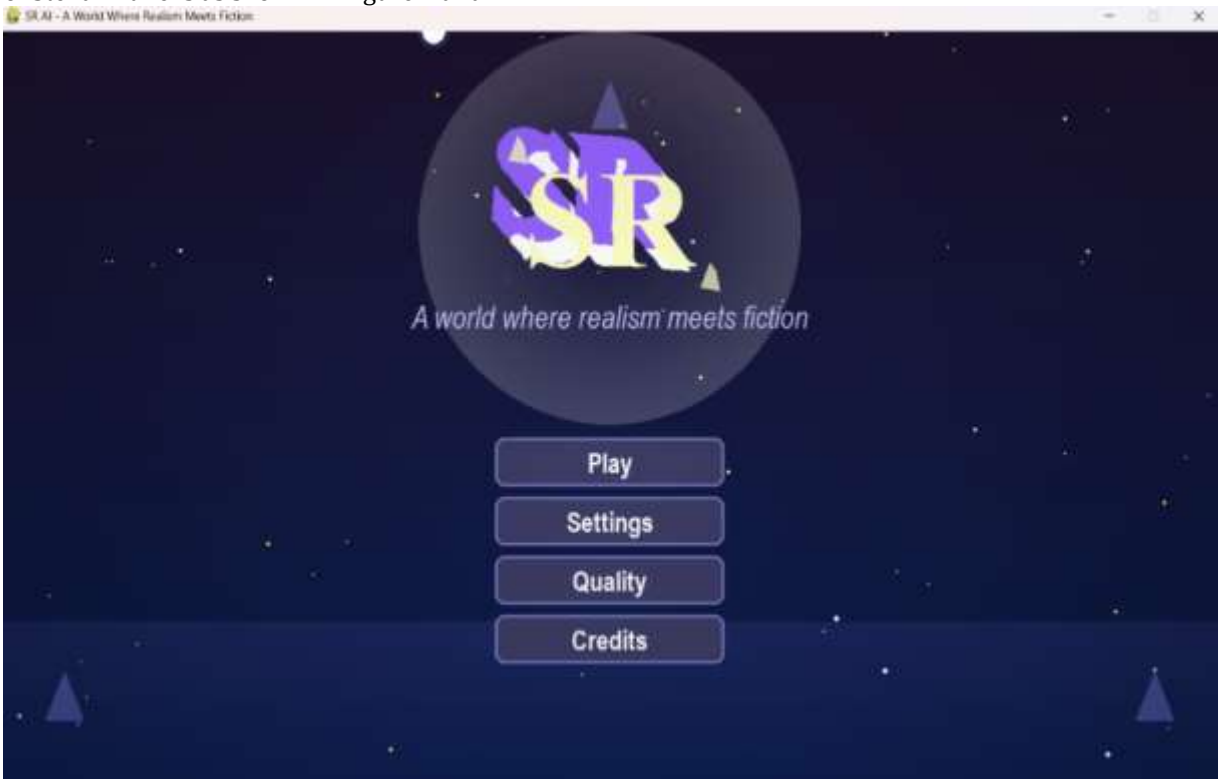

Figure 1. Stellar Rune Home Page

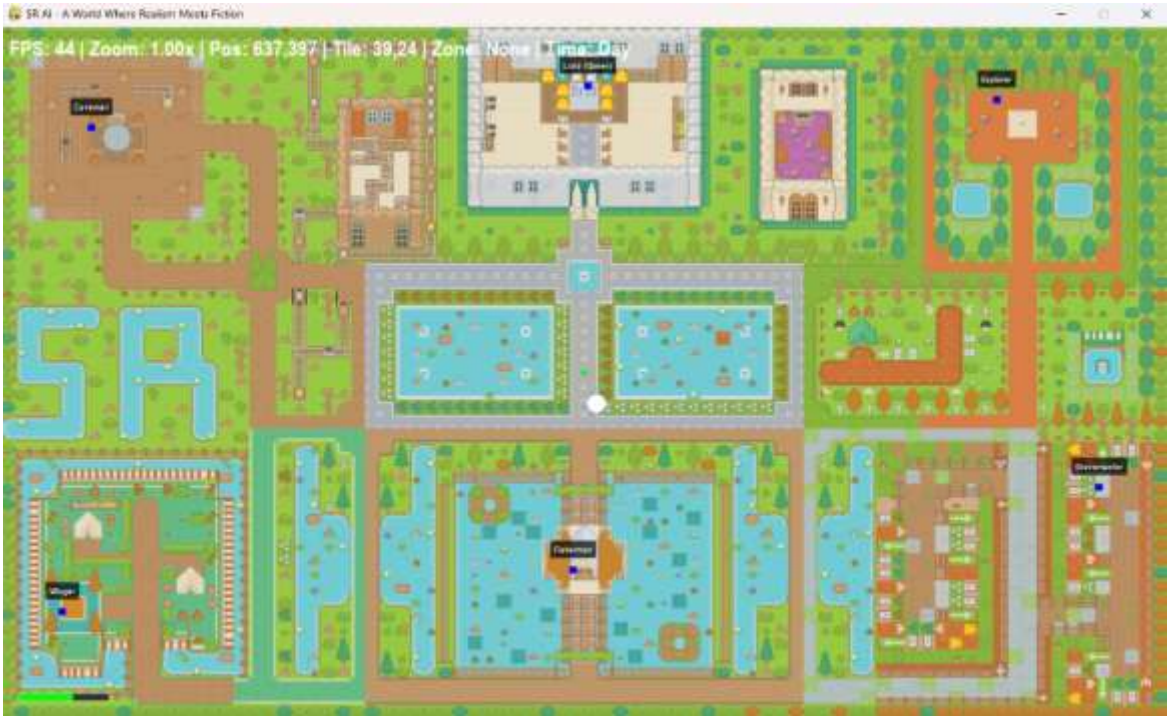

Figure 2. DGR AI gameplay system

DGR AI exists to make NPC conversations in 2D games feel less like reading a multiple-choice menu and more like actually talking to someone. Rather than focusing on graphics or mechanics, the project targets the communication layer itself: a scalable, fully offline AI that reads what the player types, classifies the intent through a two-stage neural pipeline, and generates a grounded response via a fine-tuned language model. The confidence-gated fallback and intent-conditioned generation are what separate this from a simple lookup-table chatbot. Stellar Rune is the environment where all of this runs in real time, giving the system a multi-NPC, interactive context to be tested against actual player input.

The primary goal of DGR AI is to maintain player immersion by having NPCs seem emotionally intelligent and logically consistent during a series of interactions. The NPCs, for instance, will be required to remember past player interactions, react in different ways according to time of day or game location, and even recognize patterns like frequent questions which can lead to confusion. This results in more immersive involvement and a sense of more narrative control for the player, along with the current expectations for player-led storytelling. DGR AI also enables a modular model for growth, and it becomes easy to add new intents, responses, and types of interactions. In older 2D games, communication with non-playable characters (NPCs) was quite simplistic. It is generally based on some options to speak using buttons or menus. Due to this, the dialogue can lack depth, and it's hard for players to truly become emotionally attached to the game characters. DGR AI aims to change this by allowing NPCs to understand and reply to whatever the players type, making interactions more natural and realistic. Most AI programs require an active internet connection, which is undesirable if the internet is slow or not accessible at all. DGR AI fixes this by having such intelligent conversations take place fully offline. This enables all the players to take advantage of AI features in the game regardless of their internet connectivity. DGR AI assists game developers in creating these immersive experiences. It provides NPCs with memory, knowledge of the game events, and even a personality, which makes the game world come alive and invites players to explore extensively and become emotionally invested.

2. LITERATURE SURVEY

In this section we discuss the recent research works which focuses on decentralization, memory-to-memory interactions, and the creation of context-aware systems. There has been a wave of research on smart computer systems for video games and chat platforms since 2018. Himeur, Y and et.al. [1] proposed a blockchain-based recommender system which demonstrated how a decentralized network could enhance transparency and trust when recommending content. Calvaresi and et al. [2] proposed an Agent-Based Chatbot Architecture framework which emphasized privacy and flexibility for multi-agent systems even under strict privacy regulations such as GDPR (European Union, 2016). Context-Aware Personal Assistants (2015–2021), such as those of Alexa and Mycroft, indicated the necessity to maintain conversation history [3]. During the 2020s, Transformer models became extremely popular as they were capable of comprehending language with very high accuracy [4]. Authors of [5] discusses the role of conversational AI in serious games. Aditya Mehta and et.al. [6] explores the viability of Conversational AI for Non-Playable characters. [7] presents the spoken conversational interface designed to keep users engaged in a conversation in a video game. Aqsa Sabir[8] presents an approach for the integration of conversational AI based serious games into the educational curriculum to enhance the problem-solving skills. [9] presents a AI as a visual conversational agent as a cooperative game 'GuessWhich' to measure human-AI team performance in the specific context. Cardona-Rivera, R., & Young, R. [10] discusses games as a context for communicative exchange between a game player and the game and also presents various problems within the interactive digital games community.

Research conducted in the usage of conversational AI in games from 2018 to 2025 listed static dialogue as the main concern affecting story immersion. DGR AI addresses this issue by making NPCs responsive, interactive, and context-aware, ensuring they perform well in real-time. These prior works, although varying in their methods, greatly inspired DGR -AI's design and objectives. Table. 1 presents a comparison of features of DGR AI with the a few state-of-the-art techniques discussed above.

The unique features offered by DGR AI are as given below:

1. Dynamic Thresholding Based on Input Length:

DGR AI adjusts confidence thresholds dynamically by assigning stricter thresholds to shorter inputs and more leniency to longer inputs.

Table 1 Feature comparison of DGR AI with the state-of-art techniques in the literature.

Literature Survey	Key Element	How DGR AI aligns	Uniqueness of DGR -AI
Blockchain-Based Recommender System[1] (Himeur, Y.2022)	Used blockchain for decentralization, transparency, and trust in content recommendation.	DGR AI borrows the idea of distributed decision-making, where NPC behaviours and logic are decoupled and modular.	More lightweight and easily extensible without needing blockchain overhead. Faster updates & game logic tuning.
Agent-Based Chatbot Architecture[2] (Calvaresi et al., 2023)	Focused on privacy-compliant multi-agent systems (GDPR safe), with flexible agent-based AI design.	DGR AI uses local-only processing, meaning no data leaves the system — perfect for privacy. It also allows NPCs to behave independently	Enhanced personalization of NPCs. Simpler implementation without compromising privacy.
Context-Aware Personal Assistants.[3]	Demonstrated need for conversation history & user memory (used in Alexa, Mycroft)	DGR AI implements this through its ConversationContext class: tracks recent intents, responses, and keyword patterns.	Adds context bonuses and quest-awareness, enhancing long-term coherence. Memory impacts intent scoring.
Transformer-Based NLP Models.[4]	Demonstrated that semantic em-beddings capture user meaning across diverse phrasing.	DGR AI uses Sentence-BERT (MiniLM) + cosine similarity for semantic intent detection.	Combines semantic understanding with contextual bonuses, improving robustness. Uses a hybrid model structure which boosts accuracy of results when intent return score is low.
Conversational AI in Games[5]	Demonstrated static NPC dialogues, weak immersion.	DGR AI allows NPCs to respond based on player input, recent history, and intent relevance, creating responsive and dynamic conversations.	Real-time adaptive dialogue generation. Enhanced story immersion using real-time. Intent correction and moon-themed personalization.

2. Combined Use of Rule-Based Correction, Neural Network Models and Semantic Similarity:
DGR AI blends all the three techniques using rule-based methods (e.g., "how are you"), neural network classifier and cosine similarity with Sentence Transformer. The rule-based correction component handles common inputs like greetings ("hello", "how are you"), farewells ("bye", "good night"), and basic questions ("what is your name?"), allowing quick responses without model inference.
The neural network classifier manages more complex inputs by predicting intent, while low-confidence cases are resolved using cosine similarity with Sentence-BERT to find the closest semantic match.
Memory-Constrained Local Processing:
DGR AI operates entirely on local hardware with zero cloud dependency. DGR AI explicitly avoids remote processing which is ideal for offline gaming or privacy-sensitive systems.
Real-Time NPC Adaptability for Games:
NPCs in DGR AI respond differently each time based on user input, history, and context without depending on triggers. Unlike traditional game dialogue trees, DGR AI blends NLP intelligence into in-game storytelling.
Inbuilt Model Performance Visualizer (Tkinter + Matplotlib)
DGR AI displays accuracy and loss graphs in real-time GUI tabs. The above state-of-art techniques does not implement direct integration of training metrics into the user-facing chatbot interface.
Typing Indicator & Realistic Response Delay
DGR AI introduces simulated typing animation and delays based on response length and processing time.
Context Bonus Mechanism
DGR AI assigns a mathematical bonus to intent similarity if recent user inputs align with a specific theme (e.g., quests). This fine-grained scoring approach is not present in any of the state-of-art techniques discussed above.
The proposed DGR AI uses BERT [11] and Sentence-BERT[12] models to assist with intent recognition by applying cosine similarity scoring. This feature helps DGR AI with the capability of comprehending the varied phrases people can use to describe a single notion.

3. PROPOSED TECHNIQUE

DGR AI is a sophisticated gameplay but visually simple game; it's a 2D tile-based game with graphical styles designed using Pygame and Tiled Map Editor as shown in Figure 3. It has an old pixelated graphic look to it, but the game does incorporate some newer features, like the capability to zoom in and out, particle effects, and camera fades. The proposed DGR AI game system has a very comprehensive modular, multi-layered structure for in-game and real-time gameplay interactions that require contextually sensitive give-and-takes between NPCs and players. The game system is presented by three levels of integration: the visual gameplay interface, a deep learning-based AI conversation generator, and a logic-controller layer that prescribes the state of gameplay, contextual awareness, and responses. Non-player characters are generated on the fly from the preassembled object layers accessed through the tile map, with fixed points and areas of movement. Each NPC is also programmed with associated metadata such as names, dialogue choices, and inventory which allows the conversational generator to generate context-sensitive responses.
When the player comes near an NPC and the conversation begins, the real-time system of natural language input happens. There are two steps in intent classification. Firstly, a deep learning classifier as shown in Figure 4 is built out of stacked Conv1D, BiLSTM, and Dense layers. It is trained with a JSON-entered which is a specially designed intent dataset containing all available intents. This deep learning classifier is given a tokenized, padded input text and it learns global features and local features producing the required intent. Secondly, by Sentence-BERT embeddings as shown in Figure 5 and Figure 6, the input of the user is embedded within a high-dimensional space and cosine similarity between pre-embedded intent. The model generates an output, which is validated against an 85% confidence threshold to check for correctness. If the intention classification is uncertain, an intention classification fails and triggers a second course of action that necessitates a contingency plan that involves the use of a fine-tuned language model to construct an adequate reply based on the previous conversation.

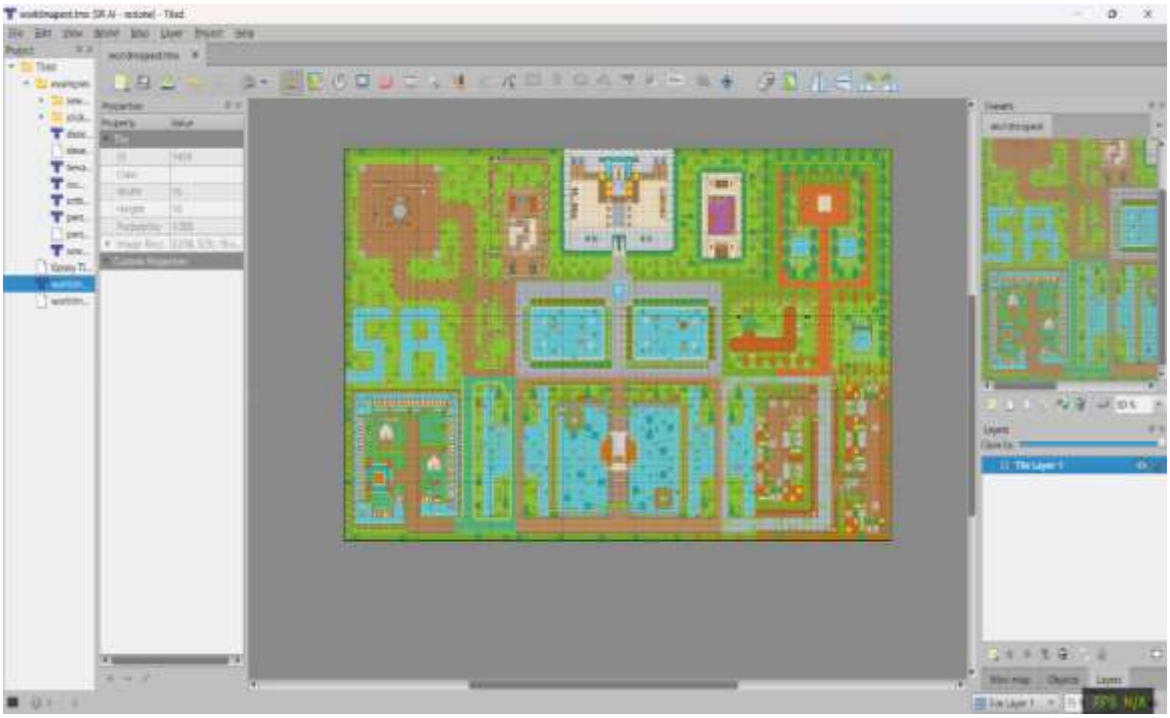

Figure 3. Tiled Application of DGR-AI

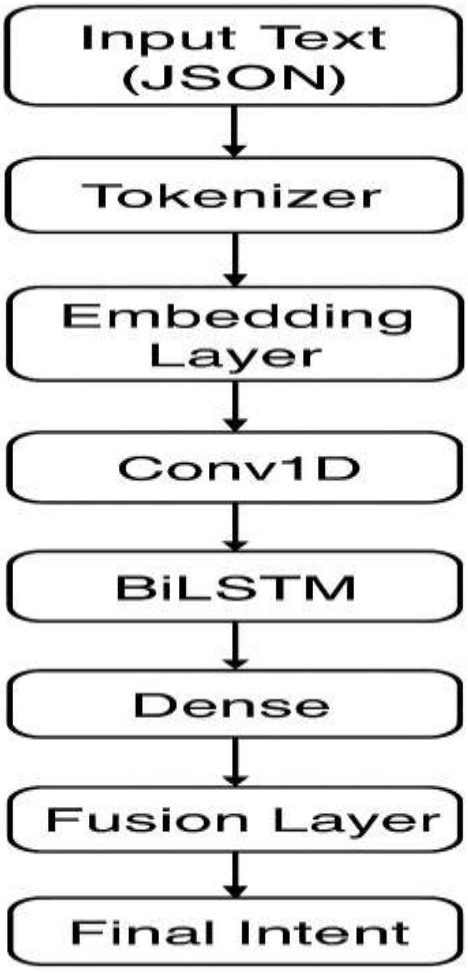

Figure 4. Hybrid Model Approach

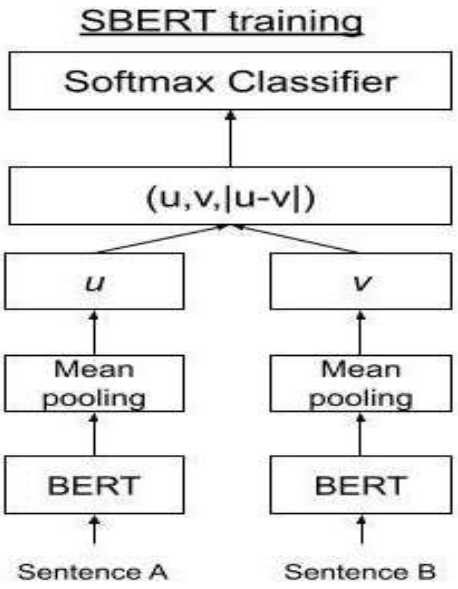

Figure 5. Sentence BERT training

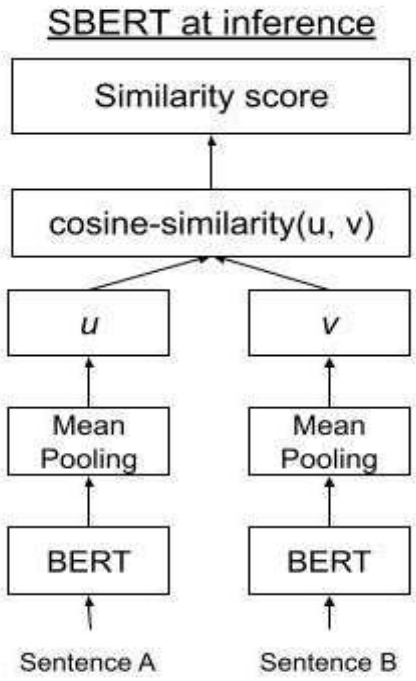

Figure 6. Sentence BERT Inference

3.1 Dataset

To maintain a smooth and efficient flow of processes, authors have proposed a custom, model-generated intent dataset which is used in training, testing and predicting the outputs of the user entered data. The intent classification system in DGR AI was trained using a custom JSON-formatted dataset, specially designed to capture a wide range of natural language expressions. As shown in Figure 7 the dataset consists of over 80 intent categories, including greetings, farewells, emotions, combat, trading, weather inquiries, and quest-related dialogue. Each intent in the dataset contains a rich set of user input patterns (phrases/questions) and a pool of corresponding NPC responses as shown in Figure 8. The dataset was manually curated to reflect realistic, context-aware conversational patterns commonly encountered in gaming environments. This allowed the deep learning model to effectively learn diverse linguistic structures and better generalize to unseen user inputs during gameplay. The dataset also supports fine-grained distinctions (e.g., between “casual greeting” and “formal greeting”), making DGR-AI’s dialogue system more immersive and responsive. This vast variety of intents for classification, makes DGR AI extremely personal and game-based, ensuring all aspects of communication are covered while preventing

overfitting with the hybrid model approach.

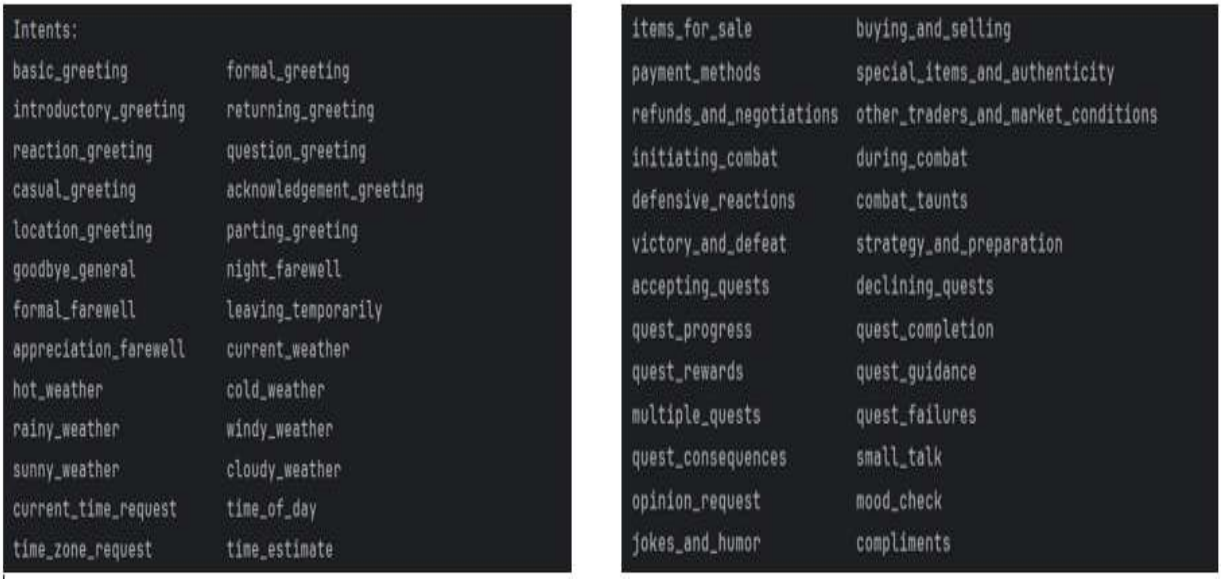

Figure 7. Example of Intents Dataset

Figure 8. Example of input patterns (questions/phrases) for intent 'question' and intent 'night farewell' and its respective responses.

3.2 BLOCK DIAGRAM/FLOWCHART DESCRIPTION OF DGR AI PIPELINE

The detailed flowchart shown in Figure 9 graphically illustrates the order of operations in the proposed DGR AI system, from getting a user input to providing the final AI-generated response. Each block represents a logical step in the intent recognition and response generation pipeline, enabling fall back-safe, solid conversational AI within the game environment.

Figure 9. Detailed block diagram of the proposed DGR AI system

Step 1. User → Input (Prompt)
The game system begins with the player or the user inputting a message or a query. This prompt forms the base of the intent detection pipeline. Figure10 shows the intents or prompts given by the user or player.

```
Input: hello | Intent: basic_greeting | Score: 0.83 | Threshold: 0.74
Input: where are you | Intent: location_inquiry | Score: 0.85 | Threshold: 0.72
Input: whats the weather like | Intent: current_weather | Score: 0.90 | Threshold: 0.71
```

Figure10. Intents or Prompts given by the User/Player.

Step 2. Randomizer Value Set
The Randomizer Value Set adds natural variation to DGR -AI's responses by selecting replies randomly from a list associated with each intent. For example, the input "hello" may trigger any of several greeting responses like "Hi!", "Hey there!", or "Good to see you!", making conversations feel less repetitive. This randomness is also applied to emotional tone, time-based responses, and NPC mood, adding subtle diversity to each interaction. It is implemented using standard randomization techniques, ensuring the dialogue remains dynamic and immersive, even when the same input is repeated.

Step 3. Dataset Load
The intent-labelled dataset is loaded into memory. This dataset contains user patterns, intents, and possible responses that the model was trained on.

Step 4. Tokenization
The input prompt is tokenized and converted into integer sequences based on a vocabulary, using a Tokenizer as given in Equation 1 and Equation 2. This makes it compatible with the model's input layer.

$x = \text{Tokenizer}(\text{sentence}) \rightarrow [w_1, w_2, \ldots, w_n]$ (1)

$e_i = \text{Embedding}(w_i) \in R^d$ (2)

where,

w_i The i-th word/token from the sentence

e_i A neural layer that converts tokens into dense vectors

x The final tokenized output (a list of token IDs or word indices)

$w_1, w_2, \ldots, w_n$ Individual tokens from the sentence (e.g., ['where', 'is', 'lake'])

$\mathbb{R}^d$ The d-dimensional real vector space (e.g., if d = 128, then $e_i \in \mathbb{R}^{128}$)

Step 5. First Intent Detection

The First Intent Detection stage uses a lightweight logic layer or a shallow model to perform a quick, initial scan of the user input. This step is optimized for speed, al-lowing DGR AI to make a rough guess of the probable intent before invoking heavier models. The first intent detection works as follows:

•Keyword matching or regex rules for commonly used phrases (e.g., "what's your name" → identity intent).

•Heuristic checks for direct command patterns or intent-specific phrases.

This allows the system to perform early overrides, such as instantly triggering known quick intents (like greetings or farewells), or routing inputs to the correct model path efficiently. It enhances overall system performance by avoiding unnecessary deep model calls for simple or easily recognizable inputs.

Step 6. Pre-processing

Further text normalization, such as lowercasing, punctuation removal, or padding/truncation, is applied to standardize inputs.

Step 7. Model Call

The trained neural network model based on Conv1D + BiLSTM + Dense is triggered to process the user input and return a predicted intent and confidence score.

Step 8. Train/Test Data Split

In DGR -AI, Train/Test Data Split refers to dividing the dataset into two parts: one for training the model and one for testing its performance. The training set is used to teach the model to recognize patterns and classify intents correctly. The test set is kept separate and is used to evaluate how well the model performs on unseen data. During live evaluation or updates, DGR AI may also simulate this split to track real-time accuracy and adjust.

Step 9. Class Weights Assigned

To counter data imbalance across intents, class weights are assigned during model training or runtime calculation, improving recognition of rarer intents as Equation 3:

$$,Weight\text{-}c. = ,N\ /\ \text{-}C.,n\text{-}c.. \quad (3)$$

where:

N= Total number of training samples

nc= Number of samples in class c

C = Total number of classes

Step 10. Neural Network Accessed

In this step, the system executes the forward pass through the model using the pre-processed input to obtain the predicted probabilities of each intent as Equation 4:

Step 10. Neural Network Accessed

$$z^1 = ReLU(W^1 \cdot h\_avg + b^1) \quad (4)$$

where,	
z^1	Output of the first dense (hidden) layer
ReLU()	Rectified Linear Unit activation function: ReLU(x) = max(0, x)
W^1	Weight matrix of the first dense layer
h_avg	Average pooled hidden vector from previous layer (e.g., embedding or LSTM)
b^1	Bias vector added in the first dense layer

b^1	Bias vector added in the first dense layer

Step 11. Second Intent Detection

In this step, the output from the model is used for a more refined second pass at intent classification as shown in Equation 5, Equation 6 and Equation 7.

$$\rightarrow h_t = LSTM\_forward(e_t, \rightarrow h_{t-1}) \quad (5)$$

$$\leftarrow h_t = LSTM\_backward(e_t, \leftarrow h_{t+1}) \quad (6)$$

$h_t = [\rightarrow h_t ; \leftarrow h_t]$ (7)

Where,	
$\rightarrow h_t$	Hidden state at time t from the forward LSTM
$\leftarrow h_t$	Hidden state at time t from the backward LSTM
e_t	Input embedding at time t (i.e., vector from the embedding layer)
$\rightarrow h_{t-1}$	Previous forward hidden state (from time t-1)
$\leftarrow h_{t+1}$	Next backward hidden state (from time t+1)
h_t	Final hidden state at time t, formed by concatenating $\rightarrow h_t$ and $\leftarrow h_t$

Step 12. Accuracy and Related Values Calculated
Metrics such as confidence score, cosine similarity, and matching accuracy are calculated to assess the confidence and accuracy of the proposed system. Intent, Confidence, Cosine Similarity and Accuracy is calculated as given in Equation 8, Equation 9, Equation 10, Equation 11.

$Intent = argmax_i(\hat{y}_i)$ (8)

$Confidence = max(\hat{y}_i)$ (9)

$CosineSim(u, v) = (u \cdot v) / (\|u\| \times \|v\|)$ (10)

$Accuracy = ,Number\ of\ Correct\ Predictions./,Total\ Predictions.$ (11)

Step 13: Final Intent Cross-checked with Override Class
The first intent is compared with a backup override system (e.g., semantic similarity or rule-based logic) to detect conflicts or refine low-confidence predictions.

Step 14. Intent Override Class Called
If there are low-confidence predictions, the override system is activated to correct or refine the initially predicted intent as given in pseudo code below:

```
input_embedding = semantic_model.encode([user_input])[0]

similarity = {
  intent: cosine_similarity([input_embedding], [embedding])[0][0]
  for intent, embedding in intent_embeddings.items()
}

best_match = max(sims.items(), key=lambda x: x[1])

if best_match[1] >= 0.8:

  final_intent = best_match[0]
else:
  final_intent = predicted_intent_from_model
```

Step 15. Final Calculated Intent Stored
Once cross-check and override step is completed, the final intent is locked in and saved for response generation.

Step 16. Confidence Check (if confidence > 0.85)
If the confidence score of the system is above 85%, it proceeds directly to the response module. Otherwise, a fallback process is triggered. It is calculated as Equation 12.

$$\mathcal{L} = -\sum(from\ i = 1\ to\ C)\ y_i \cdot log(\hat{y}_i) \quad (12)$$

Where	
$\mathcal{L}$	Total loss (cross-entropy)
C	Number of intent classes
y_i	Ground truth value for class i (1 if correct class, else 0)
$\hat{y}_i$	Predicted probability of class i (from softmax output)
log()	Natural logarithm

Each training step minimizes this loss, which helps the proposed model predict correct intents more confidently.

Step 17. Response/Output Returned
If the confidence score is high (0.85 on average) the appropriate response from the dataset is returned and shown to the player via a chat bubble or output window.

Step 18. Intent Updated
If any intermediate refinement or correction occurred, the updated intent is now finalized.

Step 19. Confidence < 0.85 → Fallback Path
If the confidence score is below 0.85, the system checks for semantic similarity or triggers a second round of evaluation.

Step 20. Second Run?
If enabled, the system re-attempts detection using a backup method.

Step 21. Override Class is selected
If the fallback doesn't help or confidence score is still low, the system selects the most probable override intent based on cosine similarity.

Step 22. Model Select → Fallback Model Called
If the primary model's prediction is low-confidence, or if the user input is ambigu-ous or unmatched, a fallback mechanism is activated.

Step 23. Intent Re-calculated
The fallback model makes a fresh prediction, or a response generation based on lan-guage understanding and not classification.

Step 24. Intent Updated
The new intent is written back to the flow, replacing the original low-confidence one.

Step 25. Output Displayed
Finally, the system displays the response to the player based on the finalized or fallback-generated intent.

4. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

This section discusses the experimental results and performance evaluation of DGR AI game system. Figure 11 shows the user/player inputting a query or a message. It also displays the intent classified by our proposed system. The confidence score and the threshold considered.

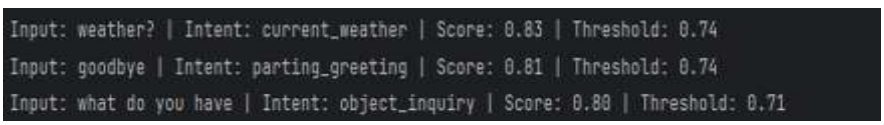

Figure11. Experimental Result of the message/ query given by the user, its intent classification, confidence score and the threshold considered.

Figure 12. Experimental Result of the player/user asking a query to the NPC in DGR -AI

Figure. 12 and Figure 13. displays the experimental results of the instances of interaction between the player and DGR AI system. In Figure 12 the player asks a query "Is it night time already" to the NPC character Luna(Queen). Figure 13 shows the reply given by the NPC character Luna(Queen) as " I think you are asking about the time of the day. Nightfall has already started." This reply validates that the proposed system correctly classifies the intent of the message and provides very appropriate answer to the query.

Figure 13. Experimental Result of the NPC replying to the query of the Player/User

4.1 Subjective Evaluation

The performance evaluation of the proposed DGR AI system is done subjectively as well as objectively. The subjective evaluation is done by set of 10 players who interacted with DGR -AI. These players were students, researchers and faculty members from the department. The average subjective evaluation scores for all the 10 players are as shown in Table 2. The subjective evaluation score included correct response by the proposed system to the query or message, the aesthetics and usability, experience, functionality, design and atmosphere of DGR -AI. The scores in Table 2 validates the efficiency of DGR AI game system in user experience.

Table 2 Subjective Evaluation of proposed DGR AI system

Player	Good	Acceptable	Bad
Player 1-10	80%	20%	0%

4.2 Objective Evaluation

The objective evaluation of the proposed system is done based on three types of different metrics used at different points of time to calculate the accuracy of the DGR AI model. Objective evaluation is done by:

- Intent Accuracy via Neural and Semantic similarity
- Predicted Accuracy via Confusion Matrix, and
- Actual Accuracy via Logs and fallback handling success.

Intent Accuracy

Intent Accuracy measures how often the proposed system correctly identifies the user's intended meaning (intent), either through the model's direct prediction or by using semantic similarity techniques like cosine similarity from sentence embeddings. Intent accuracy is calculated as Equation 13 given below.

$$Intent\ Accuracy = Correct\ NN\ Predictions + ,(Correct\ Semantic\ Matches- / Total\ Samples) \quad (13).$$

The intent accuracy for DGR AI is calculated as:

Intent Accuracy = 860 + 50 / 1000 = 0.91 = 91%

Predicted Accuracy

Predicted Accuracy measures the proportion of user inputs for which the neural network model directly predicts the correct intent label using its SoftMax output (via argmax). Predicted accuracy is calculated as Equation 14 given below.

$$Predicted\ Accuracy=Correct\ Model\ Predictions\ /, (Total\ Samples).= ,TP\text{-}NN\ ./N \quad (14)$$

The predicted accuracy for DGR AI is calculated as:

Predicted Accuracy = 860/1000 = 0.86 = 86%

Actual Accuracy

Actual Accuracy measures the percentage of user inputs for which the proposed system returns a correct or meaningful response, regardless of how the intent was identified. Accuracy is calculated as Equation 15 given below.

$$Actual\ Accuracy = (TP\_NN + TP\_semantic + ,TP\text{-}fallback.) / N \quad (15)$$

The actual accuracy of DGR AI is as given below:

Actual Accuracy = 860 + 40 + 50 /1000 = 0.95 = 95%

Table. 3 depicts objective evaluation score as the average intent accuracy, predicted accuracy and actual accuracy score of the proposed DGR AI system.

Table 3 Objective evaluation score of DGR AI system

Intent Accuracy	Predicted Accuracy	Actual Accuracy
91%	86%	95%

4.3 Performance Analysis

The balanced datasets, dropout layers, class weights, and L2 regularization helps DGR -AI's core intent classifier reach validation accuracy of upto 86%. Its hybrid Conv1D–Bidirectional LSTM architecture

captures both short and long-term dependencies in user input. A custom dataset in JSON format is loaded during runtime which consists of prompt-response pairs, essential for training and testing the hybrid

model. It aids the in-game interactions, alongside being paired with padded token sequences and error handling to strengthen the relevance. Sentence-BERT embeddings further enhance semantic understanding. During inference, cosine similarity is calculated to improve accuracy in low-confidence scenarios. The system also includes fallback logic and suggestion prompts for unclear queries, minimizing dead ends and enhancing user experience.

Performance was also validated for in-game, logging NPC state, player proximity, typing accuracy, and bubble responsiveness. Even with real-time AI processing, FPS remained consistently above 60. Offline functionality, semantic recall, and the low latency make DGR AI both powerful and player-friendly. Ultimately, DGR AI is not just a tool for interaction—it's a dynamic and evolving framework that fosters immersive and emotionally rich gaming experiences. The roadmap envisions DGR AI not just as a dialogue engine, but as a core component of immersive, emotionally intelligent, story-rich gaming environments.

5. CONCLUSIONS

The current work, DGR AI revolutionizes NPC interaction by integrating real-time conversational AI into the very centre of gameplay. It breaks away from pre-scripted dialogue, allowing responsive, context-based conversations informed by deep learning and semantic comprehension. NPCs recall, respond, and adapt—making players active contributors rather than passive observers to living narratives. Designed to perform at its best while offline with high performance and minimal latency, DGR AI demonstrates that it is not just about graphics or mechanics, but it is about games listening and responding. With its highly flexible design and emotional intelligence, DGR AI establishes a new benchmark for interactive, story-driven game development. As the videogame industry moves closer to emotionally intelligent narrative, DGR AI presents a scalable, flexible, and immersive solution. The modular design of DGR AI makes it super easy to adapt to game engines like Unity or Godot, which is great for larger projects or cross-platform releases. The subjective evaluation as well as the objective evaluation score validates proposed DGR AI as a game play system.

6. FUTURE WORK

The most significant upgrade is replacing the current incremental fine-tuning loop with a genuine reinforcement learning layer. Response selection would shift to a bandit-based approach per intent, where each response in the pool maintains a running score that updates in real-time based on a composite reward signal. The reward combines four signals: conversation continuation (player re-engages within 30s), explicit feedback (a lightweight prompt after each NPC response), task completion (player follows through on a quest or action the NPC referenced), and follow-up sentiment (the sentiment classifier scores the player's next message). This turns the system from "learns offline from stored logs" into "adapts per-interaction during play," with the selection mechanism naturally balancing exploration of new responses against exploitation of proven ones. Beyond this, swapping SBERT for XLM-RoBERT embeddings would extend the pipeline to non-English input with minimal rework, and a local STT/TTS pair (Whisper + a lightweight TTS model) would add voice I/O as a UX layer without touching the core architecture.

REFERENCES

[1] Himeur, Y., Sayed, A., Alsalemi, A., Bensaali, F., Amira, A., Varlamis, I., Eirinaki, M., Sardianos, C., & Dimitrakopoulos, G. (2022). Blockchain-based recommender systems: Applications, challenges and future opportunities. Computer Science Review, 43, 100439. https://doi.org/10.1016/j.coDGR ev.2021.100439

[2] Calvaresi, D., Eggenschwiler, S., Mualla, Y., Schumacher, M., & Calbimonte, J.-P. (2023). Exploring agent-based chatbots: A systematic literature review. Journal of Ambient Intelligence and Humanized Computing, 14, 11207–11226. https://doi.org/10.1007/s12652-023-04626-5

[3] Eduardo Islas-Cota, J. Octavio Gutierrez-Garcia, Christian O. Acosta, and Luis-Felipe Rodríguez. 2022. A systematic review of intelligent assistants. Future Gener. Comput. Syst. 128, C (Mar 2022), 45–62. https://doi.org/10.1016/j.future.2021.09.035

[4] Tucudean G, Bucos M, Dragulescu B, Caleanu CD. 2024. Natural language processing with transformers: a review. PeerJ Computer Science 10:e2222 https://doi.org/10.7717/peerj-cs.2222

[5] Göbl, B., Kriglstein, S., & Hlavacs, H. (2021). Conversational Interfaces in Serious Games: Identifying Potentials and Future Research Directions based on a Systematic Literature Review. International Conference on Computer Supported Education.

[6] A. Mehta, Y. Kunjadiya, A. Kulkarni and M. Nagar, "Exploring the viability of Conversational AI for Non-Playable Characters: A comprehensive survey," 2021 4th International Conference on Recent Trends in Computer Science and Technology (ICRTCST), Jamshedpur, India, 2022, pp. 96-102, doi: 10.1109/ICRTCST54752.2022.9782047.

[7] Jamie Fraser, Ioannis Papaioannou, and Oliver Lemon. 2018. Spoken Conversational AI in Video Games: Emotional Dialogue Management Increases User Engagement. In Proceedings of the 18th International Conference on Intelligent Virtual Agents (IVA '18). Association for Computing Machinery,

New York, NY, USA, 179–184. https://doi.org/10.1145/3267851.3267896
[8] Sabir, Aqsa. (2024). Integrating Conversational AI-Based Serious Games for Enhanced Problem-Solving Skills in Construction Education. ICCEPM 2024, The 10th International Conference on Construction Engineering and Project Management, Jul. 29-Aug.1, 2024, Sapporo.
[9] Chattopadhyay, P., Yadav, D., Prabhu, V., Chandrasekaran, A., Das, A., Lee, S., Batra, D., & Parikh, D. (2017). Evaluating Visual Conversational Agents via Cooperative Human-AI Games. Proceedings of the AAAI Conference on Human Computation and Crowdsourcing, 5(1), 2-10.
https://doi.org/10.1609/hcomp.v5i1.13312
[10] Cardona-Rivera, R., & Young, R. (2021). Games as Conversation. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 10(4), 2-8.
https://doi.org/10.1609/aiide.v10i4.12753
[11] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. North American Chapter of the Association for Computational Linguistics.
[12] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Conference on Empirical Methods in Natural Language Processing.