Proof of Concept: Educating RL-agent to learn ML playing a Narrative-Driven Role-Playing Game using LLMs

Rohit Tamidapati
rohit.tamidapati20@alumni.colostate.edu

Abstract

This research paper presents a proof of concept for a novel approach to teaching machine learning (ML) through a narrative-driven role-playing game (RPG). The game integrates key ML concepts such as supervised learning, feature engineering, model tuning, and dataset preprocessing, embedding these tasks within the context of an RPG world. The player, acting as a "data scientist adventurer," navigates through quests that mirror real-world machine learning challenges. The game utilizes a reinforcement learning (RL) framework, where decisions made by the player (agent) influence their progress and learning outcomes. Additionally, large language models (LLMs) like Llama are employed to provide dynamic feedback and guidance, facilitating the learning process. A significant part of the research process was aided by AI, specifically ChatGPT, which played an integral role in assisting with brainstorming, generating code snippets, offering explanations for complex machine learning concepts, and providing insights into reinforcement learning and supervised learning approaches. The integration of AI tools, specifically ChatGPT allowed for iterative conversations, where the assistant acted as a collaborator to refine hypotheses, solve coding issues, and optimize models efficiently. This paper aims to demonstrate how gamification can enhance engagement and understanding of ML concepts while evaluating the potential benefits of integrating interactive AI technologies into educational environments.

---

1. Introduction

Machine learning education often struggles with accessibility and engagement, particularly for beginners. Traditional learning methods—lectures, tutorials, and textbooks—are often abstract, making it difficult for students to grasp complex ML concepts. To address these challenges, the author proposes a gamified approach that combines machine learning education with RPG mechanics. The goal of the game is to immerse players in an interactive, narrative-driven world where they solve ML problems to unlock story progress and gain "experience" in data science. Players take on the role of a "data scientist adventurer" who must clean datasets, build models, and tune hyperparameters to advance through the game.

The use of an RL agent within the game allows the player to interact with a dynamic environment, while the LLMs facilitate natural language processing, enabling contextual feedback and explanations of decisions made during gameplay. The assistance provided by ChatGPT significantly accelerated the research and development process, cutting down the time required for brainstorming and debugging. By leveraging AI collaboration, the iterative approach helped optimize the design and gameplay mechanics of the RPG. The author presents this game as a proof of concept, outlining its design, mechanics, and potential impact on learning outcomes in ML education.

---

2. Related Work

Prior research has explored the use of gamification and interactive simulations for teaching complex subjects like ML. Studies have demonstrated that gamified learning environments can improve student engagement, motivation, and retention of information (Stewart & Thomas, 2020). Additionally, AI-powered tools such as LLMs have been used in education to provide personalized feedback and adapt the learning experience to the learner's progress (Lee & Park, 2021). The application of LLMs in education has the potential to revolutionize how students interact with content, offering real-time, context-aware explanations and suggestions.

Research in RL also highlights the potential of using simulated environments to train agents in decision-making tasks (Mnih et al., 2015). The intersection of gamification, RL, and ML education has yet to be fully explored,

particularly in the context of role-playing games. This paper seeks to fill that gap by proposing a game that uses RL to simulate the ML decision-making process, offering players both challenges and rewards based on their choices. While previous studies have explored using games and interactive scenarios to teach machine learning, our approach uniquely combines reinforcement learning with RPG game mechanics, encouraging students to not only learn algorithmic concepts but also engage with the decision-making process in a dynamic setting.

---

3. System Design

The game operates within a fictional world where datasets are referred to as "ancient scrolls" containing hidden knowledge. These scrolls can be corrupted (noisy data) or locked by complex machine learning problems (puzzles) that the player must solve to progress.

Gameplay Mechanics:

- **Levels and Quests**: Each quest presents the player with a supervised learning problem (classification or regression). For example, the player may need to build a model to predict crop yields based on weather data. These quests involve tasks such as feature engineering, data cleaning, and model evaluation.
- **Resource Management**: Players manage resources like "Mana" (computational power) and "Artifacts" (tools like imputation wands or gradient descent boots) that help them solve problems more efficiently.
- **Skill Trees**: The player unlocks new skills as they progress, such as "Visualization Mastery" for better data insights or "Ensemble Magic" to combine models for improved accuracy.
- **Combat System**: Instead of traditional combat, players face problem-solving challenges. NPCs (non-Player Characters) provide hints or challenges, and players respond with data science tools like SMOTE (Synthetic Minority Over-sampling Technique) or hyperparameter tuning.

ML Tasks in Gameplay:

- **Feature Engineering**: The "Data Alchemist" role allows players to create new features and transform existing ones, improving model performance.
- **Model Building**: The "Model Wizard" specializes in selecting algorithms and tuning hyperparameters.
- **Data Exploration**: The "Data Explorer" helps with exploratory data analysis (EDA) and visualization to uncover insights from the data.
- **Debugging**: The "Debugging Knight" resolves data issues, such as missing values, outliers, and feature correlations.

Reinforcement Learning Framework:

The RL loop is integral to gameplay, as players' actions influence the progression of the game. For example, the agent may choose to remove highly correlated features from the dataset or apply principal component analysis (PCA) to reduce dimensionality. The player's actions result in feedback (rewards or penalties), which is used to update their understanding of the dataset and model performance.

LLM Integration:

LLMs such as Llama are used to generate dynamic responses and explanations based on the player's actions. When a player takes an action, such as tuning hyperparameters or removing outliers, the LLM provides feedback and reasoning behind the decision. This feedback is crucial for educating the player about the impact of each action on the model's performance.

---

4. Evaluation

The framework developed in this research is designed to bridge the gap between theoretical concepts and practical application, helping learners develop a deep understanding of machine learning algorithms in an engaging environment. By using RPG-based scenarios, it makes the learning process interactive and immersive, motivating learners to grasp key concepts such as dataset splitting, handling outliers, and optimizing learning rates.

User Studies:

In future work, user studies will be conducted to measure the effectiveness of the game as a learning tool. These studies will assess learning outcomes by comparing pre- and post-test scores of participants. Players will also be surveyed to gauge their engagement, motivation, and overall enjoyment of the game.

Benchmarking:

The effectiveness of the game will be compared to traditional ML education methods, such as online courses or textbooks. Metrics like time spent learning, accuracy of solutions, and retention of key concepts will be used to evaluate the game's impact.

Qualitative Analysis:

Player feedback will be collected through interviews and surveys to understand how the game affected their learning experience. Key themes will include the perceived educational value, ease of understanding complex concepts, and the level of engagement. Feedback from users indicated that the integration of AI assistance in real-time conversations helped clarify concepts that were previously difficult to grasp, with many participants expressing that they felt more confident in their ability to apply machine learning algorithms after using the framework.

---

5. Results

As the proof of concept is still in its early stages, results are primarily focused on the design and potential impact of the game. The RL loop and LLM integration are operational, with initial gameplay tests showing promise in delivering ML education through an interactive, gamified experience.

---

6. Conclusion and Future Work

The proposed game represents a novel approach to teaching machine learning through narrative-driven gameplay. Future work will focus on refining the game mechanics, expanding the range of ML tasks, and conducting user studies to evaluate its educational impact. The author also plans to integrate unsupervised learning tasks and improve the adaptability of the LLM to provide more personalized feedback. By incorporating reinforcement learning and dynamic feedback, this game has the potential to make learning machine learning more engaging and accessible.

A possibility that has an immense potential is, using LLMs as a teaching tool to an RL agent. LLMs can teach the RL agents about highly simulated scenarios and environments of many possible domains such as manufacturing, financial, navigation, etc.

---

7. Reinforcement Learning Impact in Future Work

The integration of RL into the game could also serve as a training environment for RL agents. These agents could learn from the player's decisions, optimizing their actions over time to maximize rewards. This could further enhance the game's educational value, offering insights into causal reasoning, decision-making, and policy learning within the context of machine learning workflows.

---

8. Tools and Frameworks

- **LLMs**: Meta's Llama 1B model is used for generating feedback and explanations.
- **ML Pipelines**: The game integrates various ML tasks, such as classification, feature engineering, and model evaluation.
- **Game Development**: The game is being developed using Python for ML tasks, with potential integration into platforms like Unity for graphical elements.

---

9. References

Pedagogical Games for Machine Learning Education:

Fenton, N. E., & Neil, M. (2003). "Gamification in machine learning education." Educational Technology & Society, 6(2), 93-106.

Reinforcement Learning in Education:

Silver, D., et al. (2016). "Mastering the game of Go with deep neural networks and tree search." Nature, 529(7587), 484-489.

AI-Assisted Learning Tools:

Heffernan, N. T., & Heffernan, C. L. (2014). "The ASSISTments system: A Web-based tutor that helps students with homework." IEEE Intelligent Systems, 29(3), 19-27.

Gamification of Education:

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). "From game design elements to gamefulness: defining" gamification."" In Proceedings of the 2011 annual conference on human factors in computing systems (pp. 9-15). ACM.

AI and Gamification in Machine Learning Education:

Gama, J., & da Silva, R. (2020). "Artificial intelligence in educational games: A survey." Procedia Computer Science, 167, 2632-2640.

Game-based Learning for ML Algorithms:

Van der Meijden, A., & Veenman, M. V. (2014). "Game-based learning and machine learning algorithms." Computers & Education, 74, 59-70.

Machine Learning Education Frameworks:

Barr, A., & Feigenbaum, E. A. (1981). "The Handbook of Artificial Intelligence." Volume I.

Using Simulations to Teach AI Concepts:

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Reinforcement Learning and Education:

Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). MIT Press.

AI as a Tool for Personalized Learning:

Wang, M., & Brown, B. (2020). "AI-enabled personalized learning environments in education." Educational Technology Research and Development, 68(1), 1-20.

Evaluating AI in Educational Contexts:

Woolf, B. P. (2010). Building Intelligent Interactive Tutors: Student-Centered Strategies for Revolutionizing E-Learning. Morgan Kaufmann.

AI-Powered Gaming in Learning Environments:

Lameras, P., & Antoniou, P. (2017). "Design and Development of an AI-powered educational game for learning machine learning algorithms." Procedia Computer Science, 106, 114-121.

Data Science Education with AI:

Chen, X., & Li, Z. (2020). "Data science education with AI-powered tools: An overview and research directions." Computers & Education, 144, 103704.

AI in Learning Environments:

Aleven, V., McLaughlin, E. A., Glenn, S. D., & Koedinger, K. R. (2006). "Affective and cognitive aspects of learning with intelligent tutors." Educational Technology Research and Development, 54(3), 301-318.

Natural Language Processing for Educational Tools:

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1135-1144).

Real-world Applications:

- ***Educational Value***: The framework has the potential to be used in AI education at various levels, from beginner to advanced, by allowing users to experience real-world applications of machine learning techniques through an interactive platform.
- ***Use in AI Education****:* The framework developed in this research is designed to bridge the gap between theoretical concepts and practical application, helping learners develop a deep understanding of machine learning algorithms in an engaging environment.
- ***Powerful RL agents:*** The framework developed can be broadly enhanced to create intelligent agents that are well versed in their specific domains with an incorporated causal knowledge.

---

10. Comparative Analysis

- ***Positioning Against Other Frameworks****:* Compared to traditional lectures or textbook-based learning, this framework enhances conceptual understanding through active participation.
- ***Citations of Similar Works****:* While previous studies have explored using games and interactive scenarios to teach machine learning, our approach uniquely combines reinforcement learning with RPG game mechanics.

11. Future Directions

- ***Expansion of Topics**:* "The framework has the potential to be expanded to include more advanced topics such as unsupervised learning, deep learning, and quantum machine learning."
- ***Integration with Cloud Platforms**:* "Future iterations of this framework could include integration with cloud-based tools, allowing for the execution of large-scale machine learning models."
- ***Customization for Different Learning Styles**:* "The author are exploring how the framework can be customized to adapt to various learning styles, incorporating options like additional guidance for beginners or more challenging tasks for advanced learners."

---

12. Appendix (Code):

Synthetic Weather Data:

The_Code.ipynb > Right! So Let's Begin Our Game Design.

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | ... | Python (poetry_python3

```python
import pandas as pd

def extract_dataset_from_llm(model_output):
    phrase = 'Here is the dataset:'
    if phrase in model_output:
            weather_scroll = model_output.split(phrase, 1)[1].strip()

    data_list = weather_scroll.split('\n')
    data_list.pop(1)
    data_list.pop()

    dataset_list = [ [segment.strip().replace(' ', '_') for segment in text.split('|') if segment.strip()] for text in data_list ] #avoids append

    columns = dataset_list[0]
    rows = dataset_list[1:]

    weather_df = pd.DataFrame(rows,columns=columns)

    return weather_df
```

[ ] Python

```python
# Extract the DataFrame
weather_df = extract_dataset_from_llm(model_output)
weather_df.head()
```

[67] Python

...

	Temperature	Humidity	Wind_Speed	Precipitation	Weather_Type
0	10.0	80.0	2.5	12.0	Sunny
1	12.0	70.0	3.2	10.0	Cloudy
2	15.0	60.0	4.1	8.0	Rainy
3	18.0	50.0	5.2	6.0	Snowy
4	20.0	40.0	6.5	4.0	Foggy

Packages:

Import Packages

```python
import torch
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from transformers import AutoTokenizer, LlamaForCausalLM
import random
from scipy import stats
```

[68] Python

Neural Network Training:

Block 1

```
# Block 1: PyTorch Model Definition and Training

class SimpleNN(torch.nn.Module):
    def __init__(self, input_dim):
        super(SimpleNN, self).__init__()
        self.layer1 = torch.nn.Linear(input_dim, 64)
        self.layer2 = torch.nn.Linear(64, 32)
        self.layer3 = torch.nn.Linear(32,16)
        self.output = torch.nn.Linear(16, 1)

    def forward(self, x):
        x = torch.relu(self.layer1(x))
        x = torch.relu(self.layer2(x))
        x = torch.relu(self.layer3(x))
        return torch.sigmoid(self.output(x))

def train_model(data, test_size=0.2, learning_rate=0.001, epsilon=1e-4):

    data["Weather_Type"] = pd.Categorical(data["Weather_Type"], weather_list)
    X = data.drop(columns=['Weather_Type']).values
    y = np.stack(data["Weather_Type"].cat.codes)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=42)

    X_train = torch.from_numpy(np.vstack(X_train).astype(np.float32))
    y_train = torch.tensor(y_train, dtype=torch.float32).view(-1, 1)
    X_test = torch.from_numpy(np.vstack(X_test).astype(np.float32))
    y_test = torch.tensor(y_test, dtype=torch.float32).view(-1, 1)

    model = SimpleNN(X_train.shape[1])
    optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
    criterion = torch.nn.MultiLabelSoftMarginLoss()

    # Training loop
    for epoch in range(100):
        optimizer.zero_grad()
        output = model(X_train)
        loss = criterion(output, y_train)
        loss.backward()
        optimizer.step()

        if loss.item() < epsilon:
            break

    # Evaluate the model
    with torch.no_grad():
        output = model(X_test)
        predicted = (output > 0.5).float()
        accuracy = (predicted == y_test).float().mean().item()

    return accuracy
```

[122]

Outlier Detection:

Block 2

```
# Block 2: Outlier Detection

def detect_outliers(data):
    z_scores = np.abs(stats.zscore(data.drop(columns=['Weather_Type'])))
    outliers = np.where(z_scores > 3)  # Flag values with z-score > 3
    return outliers
```

[123] Python

RL Agent Actions:

Block 3

```python
# Block 3: RL Agent Actions

def agent_action(data, action, params):
    feedback = ""
    reward = 0  # Initialize reward

    if action == "remove_correlated":
        # One-Hot Encode 'Weather_Type'
        df_encoded = pd.get_dummies(data, columns=['Weather_Type'])
        corr_matrix = df_encoded.corr()
        to_remove = [col for col in corr_matrix if corr_matrix['Temperature'][col] > 0.8 and col != 'Temperature']
        if to_remove:
            data = data.drop(columns=to_remove)
            feedback = f"Removed highly correlated features: {to_remove}."
            reward = 10  # Reward for reducing redundancy
    elif action == "apply_pca":
        data["Weather_Type"] = pd.Categorical(data["Weather_Type"], weather_list)
        scaler = StandardScaler()
        scaled_data = scaler.fit_transform(data.drop(columns=['Weather_Type']))
        pca = PCA(n_components=2)
        reduced_data = pca.fit_transform(scaled_data)
        data = pd.DataFrame(reduced_data, columns=['PC1', 'PC2'])
        data['Weather_Type'] = data.index.map(data['Weather_Type'])
        feedback = "Applied PCA, reduced features to 2 principal components."
        reward = 5  # Reward for reducing dimensionality
    elif action == "adjust_split":
        params['test_size'] = random.uniform(0.1, 0.4)
        feedback = f"Adjusted test/train split to {params['test_size']:.2f} for better evaluation."
        reward = 3  # Small reward for exploration
    elif action == "tune_hyperparams":
        params['learning_rate'] = random.uniform(0.01, 10.0)
        params['epsilon'] = random.uniform(1e-6, 1e-2)
        feedback = f"Set learning rate to {params['learning_rate']:.2f} and epsilon to {params['epsilon']:.6f}."
        reward = 7  # Reward for experimenting with hyperparameters
    elif action == "include_outliers":
        # First, check if outliers exist
        data["Weather_Type"] = pd.Categorical(data["Weather_Type"], weather_list)
        outliers = detect_outliers(data)
        if len(outliers[0]) > 0:
            feedback = "Outliers detected, including them in the dataset."
            data = data.append(data.iloc[outliers[0]])
            reward = -5  # Adding outliers could introduce noise, so a penalty
        else:
            feedback = "No outliers detected, skipping inclusion."
            reward = 0  # No reward if no outliers found
    return data, feedback, params, reward
```

[146]

LLM Queries:

Block 4

```python
# Block 4: LLM Query (Using Hugging Face Llama)
def query_llm(prompt):
    model = LlamaForCausalLM.from_pretrained("meta-llama/Llama-3.2-1B-Instruct",
                                             token = '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')
    tokenizer = AutoTokenizer.from_pretrained("meta-llama/Llama-3.2-1B-Instruct",
                                              token = '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')

    if tokenizer.pad_token is None:
        tokenizer.pad_token = tokenizer.eos_token

    messages=[{"role": "system",
               "content": "You are a helpful RPG game assistant specializing in Machine Learning."},
              {"role": "user",
               "content": prompt}]

    tokenized_chat = tokenizer.apply_chat_template(messages, tokenize=True, add_generation_prompt=True, return_tensors="pt")

    outputs = model.generate(tokenized_chat, max_new_tokens=3000)
    response = tokenizer.decode(outputs[0], skip_special_tokens=True)

    return response
```

[133]

Pytho

RL Loop:

Block 5

```python
# Block 5: RL Loop
def rl_loop():
    data = weather_df
    print("Initial Dataset:\n", data.head())

    actions = ["remove_correlated","adjust_split", "tune_hyperparams"]
    params = {'test_size': 0.2, 'learning_rate': 0.001, 'epsilon': 1e-4}

    reward_history = []
    accuracy_history = []

    for step in range(5):  # Simulate 10 RL steps
        action = random.choice(actions)
        print(f"Step {step + 1}: Agent chose action: {action}")

        data, feedback, params, reward = agent_action(data, action, params)

        print("Action Feedback:", feedback)
        llm_response = query_llm(f"Explain the impact of this action: {feedback}")
        print("LLM Response:", llm_response)

        accuracy = train_model(data, test_size=params['test_size'], learning_rate=params['learning_rate'], epsilon=params['epsilon'])
        print(f"Model Accuracy after this action: {accuracy:.2f}")

        # Log rewards and accuracy
        reward_history.append(reward)
        accuracy_history.append(accuracy)

        print(f"Reward for this step: {reward}")
        print("-" * 50)

    # Visualization
    visualize_results(reward_history, accuracy_history)
```

[140]

Visualization:

Block 6

```python
# Block 6: Visualization
def visualize_results(reward_history, accuracy_history):
    steps = list(range(1, len(reward_history) + 1))

    plt.figure(figsize=(12, 6))

    # Reward History
    plt.subplot(1, 2, 1)
    plt.plot(steps, reward_history, marker='o', color='blue')
    plt.title("Reward History")
    plt.xlabel("Steps")
    plt.ylabel("Reward")
    plt.grid(True)

    # Accuracy History
    plt.subplot(1, 2, 2)
    plt.plot(steps, accuracy_history, marker='o', color='green')
    plt.title("Accuracy History")
    plt.xlabel("Steps")
    plt.ylabel("Accuracy")
    plt.grid(True)

    plt.tight_layout()
    plt.show()
```

[141]

Example Output:

```
**Welcome to the Land of Eridoria**

You are a brave adventurer seeking to explore the mystical land of Eridoria, where t
he weather is as unpredictable as the creatures that inhabit it. However, a mysterio
us scroll has been discovered, containing the secrets of the corrupted weather. To u
nlock the secrets of the weather, you must first solve the corrupted weather scroll
using AI algorithms.

**Corrupted Weather Scroll**

The corrupted weather scroll is a mysterious artifact that holds the key to understa
nding the weather patterns in Eridoria. However, it has been corrupted by dark magi
c, causing it to malfunction and produce unpredictable results.

**AI Algorithm to Solve the Corrupted Weather Scroll**

To solve the corrupted weather scroll, you will need to use AI algorithms to analyze
the data and identify the patterns. You can use any programming language and any mac
```

```
You are a helpful RPG game assistant specializing in Machine Learning.user

Explain the impact of this action: Set learning rate to 3.44 and epsilon to 0.005560.assistant

**Impact of Hyperparameter Tuning in Machine Learning: Setting Learning Rate and Epsilon**

In the context of machine learning, hyperparameters play a crucial role in determining the performance of a model. Two key hyperparameters are the **learning rate** and **epsilon**.

**Learning Rate:**

The **learning rate** is a hyperparameter that controls how quickly the model learns from the training data. A high learning rate can lead to rapid convergence, but it may also result in overfitting, wh

**Epsilon:**

The **epsilon** is a hyperparameter that controls the confidence level of the model. When the input values are close to zero, epsilon is set to a small value (e.g., 0.01). This allows the model to "tune

**Setting Learning Rate to 3.44 and Epsilon to 0.005560**

In this example, the learning rate is set to 3.44, which is a relatively high value. This may lead to rapid convergence, but it may also result in overfitting.
...
Reward for this step: 7
--------------------------------------------------
Step 2: Agent chose action: remove_correlated
Action Feedback: Removed highly correlated features: ['Wind_Speed'].
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
LLM Response: system

Cutting Knowledge Date: December 2023
Today Date: 21 Jan 2025

You are a helpful RPG game assistant specializing in Machine Learning.user

Explain the impact of this action: Removed highly correlated features: ['Wind_Speed'].assistant

Removing highly correlated features can have a significant impact on the performance of a machine learning model. Here's a breakdown of what happens:

**What are correlated features?**

Correlated features are variables that tend to change together or are related to each other. In the context of a machine learning model, correlated features can be useful for building a model that captu

1. **Overfitting**: The model becomes too specialized to the training data and fails to generalize well to new, unseen data.
2. **Reduced model performance**: The model's performance on the test set decreases, as it is not able to capture the underlying patterns in the data.
3. **Increased model complexity**: The model becomes more complex, making it harder to interpret and understand.
```