algorithms

MDPI

Article

Interpretable Reinforcement Learning for Sequential Strategy Prediction in Language-Based Games

Jun Zhao [1,†], Jintian Ji [2,†], Robail Yasrab [3], Shuxin Wang [2], Liang Yu [1,*] and Lingzhen Zhao [1,*]

[1] Department of Public Foundation, Wannan Medical College, Wuhu 241000, China; zhaojun@wnmc.edu.cn
[2] Department of Medical Information, Wannan Medical College, Wuhu 241000, China; 22116030156@stu.wnmc.edu.cn (J.J.); 23116020116@stu.wnmc.edu.cn (S.W.)
[3] MRC Biostatistics Unit, University of Cambridge, Cambridge CB2 1TN, UK; robail.yasrab@mrc-bsu.cam.ac.uk
* Correspondence: wytwyl@wnmc.edu.cn (L.Y.); moying2023@126.com (L.Z.)
† These authors contributed equally to this work.

Abstract

Accurate and interpretable prediction plays a vital role in natural language processing (NLP) tasks, particularly for enhancing user trust and model transparency. However, existing models often struggle with poor adaptability and limited interpretability when applied to dynamic language prediction tasks such as **Wordle**. To address these challenges, this study proposes an interpretable reinforcement learning framework based on an Enhanced Deep Deterministic Policy Gradient (Enhanced-DDPG) algorithm. By leveraging a custom simulation environment and integrating key linguistic features word frequency, letter frequency, and repeated letter patterns (rep) the model dynamically predicts the number of attempts needed to solve **Wordle** puzzles. Experimental results demonstrate that Enhanced-DDPG outperforms traditional methods such as Random Forest Regression (RFR), XGBoost, LightGBM, METRA, and SQIRL in terms of both prediction accuracy (MSE = 0.0134, R^2 = 0.8439) and robustness under noisy conditions. Furthermore, SHapley Additive exPlanations (SHAP) are employed to interpret the model's decision process, revealing that repeated letter patterns significantly influence low-attempt predictions, while word and letter frequencies are more relevant for higher attempt scenarios. This research highlights the potential of combining interpretable artificial intelligence (I-AI) and reinforcement learning to develop robust, transparent, and high-performance NLP prediction systems for real-world applications.

Keywords: interpretable artificial intelligence; reinforcement learning; natural language processing; enhanced deep deterministic policy gradient; **Wordle** prediction

Academic Editors: Affan Yasin, Javed Ali Khan and Lijie Wen

Received: 23 May 2025
Revised: 21 June 2025
Accepted: 8 July 2025
Published: 11 July 2025

Citation: Zhao, J.; Ji, J.; Yasrab, R.; Wang, S.; Yu, L.; Zhao, L. Interpretable Reinforcement Learning for Sequential Strategy Prediction in Language-Based Games. *Algorithms* **2025**, *18*, 427. https://doi.org/10.3390/a18070427

1. Introduction

Natural language processing (NLP), a core area of artificial intelligence (AI), aims to enable machines to understand, interpret, and generate human language. With the rise of intelligent systems, NLP techniques have been widely applied in diverse fields such as text mining, machine translation, and sentiment analysis. Recently, the increasing popularity of interactive word games like **Wordle** has introduced a novel and engaging scenario for studying language-based user behavior models in dynamic and gamification environments.

In **Wordle**, players attempt to guess a hidden five-letter word within six tries, receiving structured feedback after each guess. Beyond simple vocabulary recall, the game requires logical reasoning and decision-making under uncertainty. Moreover, **Wordle** publicly releases daily statistics on the number of attempts users need for different words.

These data offer rich linguistic and behavioral signals such as word frequency, letter occurrence, and repetition patterns. which provide a valuable foundation for modeling user guessing behaviors.

However, traditional machine learning methods face several challenges when modeling this problem. Although Random Forest Regression (RFR) [1] has been widely used for feature importance analysis, its complex ensemble structure often leads to overfitting, especially in noisy or high-dimensional datasets. Moreover, the prediction speed of RFR is limited due to the need to traverse multiple decision trees [2].

XGBoost [3] improves the limitations of RFR by constructing weak learners in sequence and introducing regularization to prevent overfitting. However, it requires the manual encoding of categorical features and may struggle to adapt quickly to dynamic changes in the game environment [4]. LightGBM [5] further optimizes model efficiency by adopting a leaf-wise growth strategy and supporting the native processing of categorical features. Nevertheless, it remains sensitive to noisy data and outliers, which affects its robustness [6]. DDPG [7] introduces deterministic policy gradients, avoiding the costly sampling of policy distributions in continuous-action spaces by directly updating policy parameters via gradients, which enables it to excel on high-dimensional continuous-control problems. It combines a deep Q-network with a deterministic policy network using a replay buffer and target network mechanisms, ensuring stable training while improving sample efficiency [8–10].

While deep learning has achieved remarkable success in various NLP tasks, it may not be ideally suited for this problem. First, the **Wordle** dataset is relatively small and consists of low-dimensional, well-structured linguistic features, which limits the effectiveness of deep models and increases the risk of overfitting. Second, the "black-box" nature of deep neural networks makes it difficult to explain and trust their predictions—an essential requirement when modeling human decision behavior. Third, most deep learning models are designed for static inference tasks and lack the inherent ability to interact with dynamic feedback environments like **Wordle**. These limitations further highlight the need for a method that is both adaptive and interpretable.

To address the aforementioned challenges and leverage the strengths of the original DDPG, this study proposes an integrated framework combining reinforcement learning (RL) and interpretable artificial intelligence (I-AI). Specifically, we propose an Enhanced-DDPG algorithm that dynamically predicts the distribution of **Wordle** attempt counts and incorporates SHapley Additive exPlanations (SHAP) to interpret the model's internal decision-making process.

By using key linguistic features such as word frequency, letter frequency, and repetition patterns, our approach demonstrates superior prediction accuracy and robustness compared to baseline models such as Random Forest, XGBoost, and LightGBM. Furthermore, the SHAP-based interpretability analysis reveals how different features contribute to predictions under various attempt scenarios, offering valuable insights into user behavior modeling. The main contributions of this study are as follows:

1. We propose a unified framework that combines NLP, RL, and I-AI to predict user attempt distributions in **Wordle**, achieving both high performance and transparency.
2. A feature-driven prediction model is constructed using key linguistic indicators (e.g., letter frequency, word frequency, repetition pattern).
3. A DDPG-based RL algorithm is adopted to dynamically optimize the prediction policy through environmental interaction.
4. SHAP is introduced to explain the model's predictions, uncovering how different features influence attempt classifications under varying conditions.

This study not only offers a novel solution for **Wordle**-style prediction but also provides a reference for applying I-AI in broader domains such as medical diagnosis and financial forecasting.

2. Related Works

In recent years, the intersection of NLP, RL, and I-AI has become an important driving force for the development of multiple fields. This research not only accelerates technological progress but also provides new perspectives for solving complex decision-making problems.

To address the challenges of function approximation error and overestimation bias in traditional algorithms, several improvements have been proposed. TD3 greatly enhances training stability on high-dimensional continuous-control tasks by introducing clipped double Q-learning and delayed policy updates [8–13]. Soft Actor-Critic builds on the same off-policy framework but adds an entropy regularization objective to balance exploration and exploitation and further improve sample efficiency [11]. However, neither approach fully reconciles simplicity with richer abstractions or exploration strategies.

To overcome these limitations, METRA [12] uses a learned-state distance metric derived from self-supervision or intrinsic environmental features to group similar states into abstract meta-states via clustering. It then constructs a reduced Markov decision process over these meta-states for fast planning and learning, but there is a cost of sensitivity to metric quality and cluster boundaries. SQIRL [10–13] begins with pure random or lightly greedy exploration to gather a wide diversity of experience, stores it in a replay buffer, and periodically performs multiple Bellman backups on the same batch of samples to stabilize Q-value estimates. Despite these advances, it still falls short of fully addressing sample efficiency and the challenges posed by very high-dimensional action spaces.

Compared to alternative methods like METRA or SQIRL [12,13], DDPG offers superior sample efficiency and direct policy optimization in continuous control. Though adversarial approaches (e.g., QARL) [14] enhance robustness through curriculum learning, the simplicity and compatibility of DDPG with SHAP-based interpretability make it a natural choice for our framework [15–20]. Moreover, the adaptability of DDPG in sparse-reward scenarios ensures effectiveness in real-world NLP tasks where feedback signals are often limited.

I-AI refers to AI systems that are able to provide human-understandable explanations or evidence to support their decisions. The core goal of I-AI is to improve the transparency and user trust of the model, so that its decision-making process is clear and understandable to humans. Muschalik et al. [21] propose SHAP, an artificial intelligence explanatory tool, to be applied to the decision analysis of machine learning models. SHAP provides an intuitive explanatory framework for complex black-box models by calculating the marginal contribution of each feature to the model output [22,23] and is a model-agnostic method [24,25] that can be applied to any machine learning model, be it a linear model or a complex black-box model. This universality is also the reason why SHAP outperforms other methods; it provides both local explanations, which explain how a single prediction was made, and global explanations, which understand the model's behavior as a whole [26–28]. Quantifying the importance of each feature [21,29–31], Yang et al. [32] propose a medical, explainable AI method based on multi-modal and multi-center data fusion and demonstrate its important role in revealing the black-box decision process of deep neural networks. This method simultaneously constructs a diagnostic module and an explanation module for the prediction task.

In the interpretation module, Class Activation Mapping (CAM) is used to generate intuitive heat maps that highlight the key regions the model focuses on, improving accuracy in disease classification and lesion segmentation. Baehrens et al. [33] introduce a framework for explaining individual classification decisions via local explanation vectors, addressing the "black-box" nature of machine learning models. Adebayo et al. [34] propose a randomized test evaluation including model parameter and data randomization tests to assess attention-map sensitivity, showing that gradient and GradCAM methods are sensitive to both model parameters and data generation, whereas Guided BackProp and Guided GradCAM are not. Ribeiro et al. [35] develop Local Interpretable Model-agnostic Explanations (LIMEs), which samples neighborhood perturbations and fits a linear model to reveal each feature contribution, thereby enhancing interpretability and user trust. Teso et al. [36] present Explanatory Interactive Learning (XIL), a reinforcement-learning approach that jointly optimizes predictions and local explanations, allowing users to iteratively refine attribution maps for model debugging and correction.

In a decision-making scenario, an agent combining NLP and RL techniques can optimize the policy in real time and explain its decision logic through I-AI techniques. Such application research opens new possibilities for transparent, efficient, and intelligent AI systems in text prediction [16].

Based on the theoretical models and empirical cases discussed above, it is easy to conclude that the existing research provides a strong theoretical basis and technical support for this study. However, how to achieve efficient, transparent, and robust models in dynamic and complex environments is still a challenging problem. In this study, we combine NLP, I-AI, and RL techniques to explore the possibility of achieving this goal in **Wordle** program.

3. Methods

3.1. Definition

Let $\mathcal{S}$ denote the set of state space, which includes all possible states that an agent can observe. Let $\mathcal{A}$ represent the set of action space, representing all possible actions the agent can perform. In continuous-action control tasks, $\mathcal{A}$ typically refers to a continuous vector space. Let $\mu_\theta(s)$ denote the actor network with parameters θ, which maps a state $s \in S$ to an action $a \in \mathcal{A}$. Similarly, $\mu_{\theta'}(s)$ represents the target actor network with parameters θ'. The target actor shares the same architecture as the main actor but is update via soft updates or periodic copying to improve training stability. Let $Q^{\mu}_{\omega}(s,\ a)$ denote the critic network with parameter ω, which estimates the Q-value for a state-action pair $(s,\ a)$. The critic takes both the current state s and action a as inputs and outputs an estimate value. $Q^{\mu}_{\omega'}(s,\ a)$ denotes the target critic network with parameters ω', which has the same structure as Q^{μ}_{ω} and is update via soft or periodic updates. M represents the replay buffer. This buffer is use for sampling experiences during training to reduce temporal correlation and stabilize updates. Let $\gamma \in (0, 1]$ denote the discount factor, which controls the trade-off between immediate and future rewards. A higher γ places more weight on long-term rewards. Let N represent the noise model use to promote exploration. Noise is added to the actions selected by the actor network, typically using Ornstein–Uhlenbeck or Gaussian processes during training. Let $\tau \in (0, 1]$ denote the target update coefficient for soft updating the target networks. The soft update rule is Equation (1):

$$\begin{cases} \theta' \leftarrow \tau\theta + (1-\tau)\theta' \\ \omega' \leftarrow \tau\omega + (1-\tau)\omega' \end{cases} \tag{1}$$

when τ is very small, the target network updates relatively slowly; alternatively, a "hard" update can be used, including every fixed number of steps θ' and steps ω'. Finally, the target Q-value y_i used for critic updates is computed as Equation (2):

$$y_i = r_i + \gamma Q_{\omega'}^{\mu_{\theta'}}(s_{i+1},\ \mu_{\theta'}(s_{i+1})). \tag{2}$$

3.2. Model Structure

Enhanced-DDPG is a model-free reinforcement learning method for continuous-action spaces [7–12]. By combining deterministic policy gradient and deep learning techniques, Enhanced-DDPG performs well in solving continuous-control problems. The following is the algorithm flow of our Enhanced-DDPG:

This approach aligns with reinforcement learning principles where action spaces are dynamically constrained by environmental states.

Firstly, the parameters of the critic network and actor network are randomly initialized. Specifically, the parameters φ of the critic network $Q(S, A; \varphi)$ and the parameters θ of the actor network $\pi(S; \theta)$ are initialized with random values. Furthermore, the parameters of the target critic network φ_t are set to be equal to φ. Similarly, the parameters of the target actor network θ_t are initialized to match θ.

Additionally, Enhanced-DDPG creates a cyclic experience replay buffer to store and reuse past experiences, which improves the efficiency of learning by breaking the temporal correlations in the data.

During the training phase, the Enhanced-DDPG algorithm follows an iterative process. For the current observed state S, an action A is selected using the actor network. To encourage exploration, random noise from a predefined noise model is added to the action. The selected action A is then executed in the environment, which returns a corresponding reward R_i and the next state S'. The experience tuple $(S,\ A,\ R,\ S')$ is stored in the replay buffer for later use in batch updates.

To update the network parameters, the algorithm randomly samples a mini-batch of experiences $(S,\ A,\ R,\ S')$ from the replay buffer. For each sampled experience, the algorithm computes a target value y_i. If S_i' is a terminal state, then y_i is simply set to R_i. Otherwise, the next action is computed using the target actor network, and its value is evaluated using the target critic network, leading to the following target computation as Equation (3), where γ is the discount factor:

$$y_i = R_i + \gamma Q_t\left(S_i',\ \pi_t\left(S_i'; \theta_t\right); \varphi_t\right). \tag{3}$$

The parameter is updated by minimizing the critic network of all sampling experiences with loss function L, adopting the method of gradient descent. Meanwhile, the parameters of the actor network are updated based on the sampled policy gradient to maximize the expected discounted cumulative long-term reward. The process involves calculating the gradient of the critic's network output to the action, and the gradient of the actor's network output to the parameter.

The key step of Enhanced-DDPG algorithm is to update the target network, which plays an important role in improving the stability of learning. Enhanced-DDPG supports three target update methods, including smooth update, periodic update, and periodic smooth update. In smooth update, the target parameters are updated at each time step using a smoothing factor τ. Periodic updates involve the target parameters being updated after a fixed interval without smoothing (τ = 1). The periodic smooth update combines these two methods, updating after a fixed interval using a smoothing factor.

Through this iterative training process, Enhanced-DDPG algorithm continuously optimizes the actor and critic network to achieve effective policy learning in the continuous-action space. Its advantage lies in its continuous control that can handle the high dimension problem, and by introducing the target network, effectively solve the stability problem in the process of learning. The Enhanced-DDPG algorithm and its variants continue to play an important role with the deepening of the application of reinforcement learning in automatic decision-making, autonomous driving, and other fields. For this study, the model structure is shown in Figure 1 and the model algorithm process is shown in Algorithm 1. The following is a detailed description of the main steps in the algorithm table:

Algorithm 1 The algorithm of the proposed Enhanced-DDPG.

Require :
Observation space S, *action space* A
Actor network $\pi(S;\theta)$, *target actor network* $\pi_t(S;\theta_t)$
Critic network $Q(S,A;\varphi)$, *target critic network* $Q_t(S,A;\varphi_t)$
Experience buffer size M, *discount factor* γ, *noise model* $\mathcal{N}$
Target update method(*smoothing factor* τ *or update frequency*)
Ensure :
Trained actor network$\pi(S;\theta)$
Initialize critic $Q(S,A;\varphi)$ *with random parameters* φ
$\varphi_t \leftarrow \varphi$
Initialize actor $\pi(S;\theta)$ *with random parameters* θ
$\theta_t \leftarrow \theta$
Initialize experience buffer with capacity M
for each training time step do
  Observe current state S
  $A = \pi(S;\theta) + \mathcal{N}$
  Execute action A, *observe reward* R*and next state* S'
  Store experience (S,A,R,S') *in the buffer*
  Sample a mini $-$ *batch of* M *experiences* (S_i, A_i, R_i, S_i')
  for each experience in the mini $-$ *batch do*
    if S_i' *is a terminal state then*
      $y_i = R_i$
    else
      $y_i = R_i + \gamma Q_t(S_i', \pi(S_i';\theta_t);\varphi_t)$
    end if
  end for
  Update critic parameters φ *by minimizing the loss L across all ampled experiences*
  Compute gradients $\nabla_A Q(S_i, A_i;\varphi)$ *and* $\nabla_\theta \pi(S_i;\theta)$
  Update actor parameters θ *via sampled policy gradient*
  if smoothing then
    $\theta_t \leftarrow \tau\theta + (1-\tau)\theta_t$
    $\varphi_t \leftarrow \tau\varphi + (1-\tau)\varphi_t$
  elseif periodic then
    Every k steps : $\theta_t \leftarrow \theta$, $\varphi_t \leftarrow \varphi$
  end if
end for

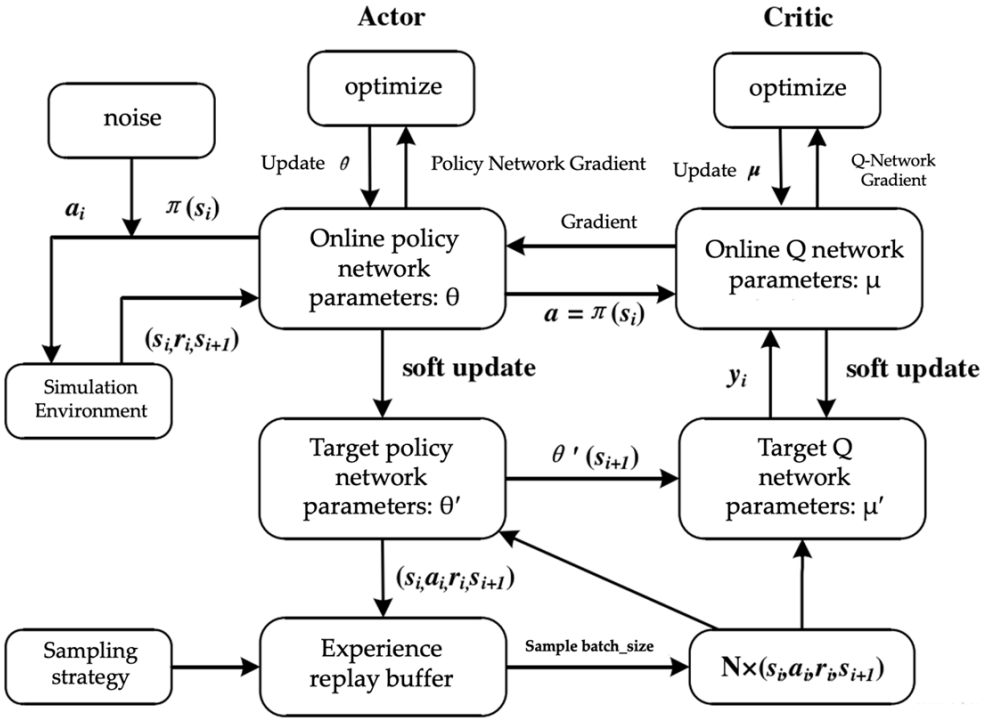

Figure 1. The workflow of Enhanced-DDPG algorithm.

First, the parameters of two neural networks (i.e., the parameters ω of the critic network $Q_\omega(s,a)$ and the parameters θ of the actor network $\mu_\theta(s)$) are randomly initialized to speed up convergence.

Next, to maintain consistent targets during training, we create target networks $Q_{\omega'}$ and $\mu_{\theta'}$ with the same architecture as the main networks and initialize them by copying the current parameters from the main networks.

Then, we construct an experience replay buffer D with capacity M to store the agent's experiences. When the buffer reaches its limit, it removes the oldest samples using a FIFO strategy to maintain diversity and timeliness.

In each training step, the actor network outputs an action a_t based on the current state s_t, which is then perturbed by noise from a predefined model to produce $a_t^{explore}$, encouraging sufficient exploration.

After executing $a_t^{explore}$, the environment returns the immediate reward r_t and next state s_{t+1}. The experience $\left(s_t, a_t^{explore}, r_t, s_{t+1}\right)$ is stored in the replay buffer for future training.

During the update phase, a mini-batch of samples is randomly drawn from the buffer. The target value as Equation (4) is computed using the target networks, where γ is the discount factor:

$$y_i = r_i + \gamma \cdot Q_{\omega'}(s_{i+1}, \mu_{\theta'}(s_{i+1})). \tag{4}$$

The critic network is updated by minimizing the mean squared error loss L, with actor parameters fixed and only critic parameters ω updated via gradient descent to make estimated Q-values closer to targets.

After update the critic, the actor parameters θ are updated using policy gradients. The gradient is estimated by the product of $\nabla_a Q_\omega(s,a)|_{a=\mu_\theta(s)}$ and $\nabla_\theta \mu_\theta(s)$, with adjustments via backpropagation.

3.3. Decision Logic and Reward Function

Based on the Enhanced-DDPG framework, the model estimates the state-action value $Q(s, a)$ using the critic network and selects optimal actions through an ϵ-greedy strategy to balance exploration and exploitation (with ϵ initial set to 0.3 and gradually decaying to 0.05). The model selects the action that maximizes $Q(s, a)$ within the valid action subset A_{valid}, as defined in Equation (5):

$$a^* = arg \underset{a \in A_{valid}}{max} Q(s, a). \tag{5}$$

The model prioritizes experience replay based on the absolute values of rewards, accelerating convergence by focusing on critical patterns (e.g., high-reward or high-penalty samples).

Additionally, SHAP values are employed to quantify the contribution of each state feature to the model's prediction, enabling the distribution of the global reward across individual feature dimensions. A novel reward function is proposed, as shown in Equation (7). During actor network updates, SHAP values constrain the policy gradient direction, ensuring that critical features influence action selection as intended.

The original reward $r(s_t, a_t)$ represents the traditional immediate reward returned by the environment after executing an action in state s_t. Then, $r(s_t, a_t)$ incorporates both positive rewards and negative penalties, where each newly revealed correct letter yields a reward of one point, correctly guessing the target word yields ten points, guessing a word not in the dictionary incurs a two-point penalty, reusing a previously attempted word incurs a one-point penalty, and making a guess that provides no new clues incurs a 0.5-point penalty.

SHAP values quantify feature importance by calculating the marginal contribution of each feature, following the Shapley value concept from game theory. Specifically, for a target sample x_{target}, the baseline prediction value $E[f(X)]$ is calculated from the background dataset. Then, for all feature subsets S, perturbed inputs are generated. The SHAP value for feature i is obtained by comparing the prediction differences when including or excluding i , followed by a weighted summation, as shown in Equation (6):

$$\phi_i = \sum_{S \subseteq X \setminus \{i\}} \frac{(|S|)!(n - |S| - 1)!}{n!} [f(S \cup \{\mathrm{i}\}) - f(S)], \tag{6}$$

where ϕ_i represents the SHAP value of feature i, S denotes feature subsets excluding i, n is the total number of features, and $f(S)$ is the model prediction based on subset S.

The proposed reward function includes a SHAP compensation term $\sum_i SHAP_i(s_t)$, which quantifies the contribution of each feature to the model's prediction. An adjustment parameter λ balances the SHAP compensation term with the original reward to achieve interpretability objectives, as shown in Equation (7):

$$R_i = r(s_t,\ a_t) + \lambda \cdot \sum_i SHAP_i(s_t). \tag{7}$$

Finally, soft updates are applied to blend the main network parameters into the target networks, ensuring smooth target tracking. This training process repeats until the policy converges to approximately optimal actions.

4. Experiments

4.1. Data Preparation and Preprocessing

We use the dataset provided openly by COMAP. To prevent large-scale features from dominating and small-scale features from being neglected, normalization is applied to address differences in feature dimensions and scales. In order to avoid such problems, we use the normalized processing of the data MinMax function and normalization processing during the process of training to improve the numerical stability. We prevent the characteristics of the larger dominant learning process, and the normalized function is shown in Equation (8):

$$V_{\text{norm}} = \frac{V - V_{min}}{V_{max} - V_{min}}, \tag{8}$$

where V_{min} is the minimum value, V_{max} is the maximum value, and V_{norm} is the value of after normalization in the range [0, 1]. We develop a set of Python 3.8-based automation processing procedures, and to analyze the frequency of the word frequency and mode double letters, we import the requests for sending HTTP requests, use pandas library for processing form data, and employ collections.Counter to count the number of occurrences of letters. We use two functions, get_word_score and get_letter_score, to query the relevance score of words and letters, respectively. These functions calculate scores by calling Datamuse API, indirectly reflecting the use frequency of word frequency and letters.

We obtain dataset words after the word frequency and the frequency of letters. Through regression analysis, we found that the data points in the low-frequency region are relatively sparse, while in the high-frequency region, they are very dense, as shown in Figure 2. This indicates the dominance of high-frequency words in the dataset, and the Tropic of Cancer negative slope indicates that the score is slightly lower than the low frequency of high-frequency words. This is associated with a certain language phenomenon. High-frequency words may be more common but have fuzzier semantics. At the same time, we define the calculate_rep function to detect whether there are repeated letters in the word. The function uses counter statistics for the occurrence of each letter in the word, and if there are any repeat letters, it returns 1.5; otherwise, it returns 0.

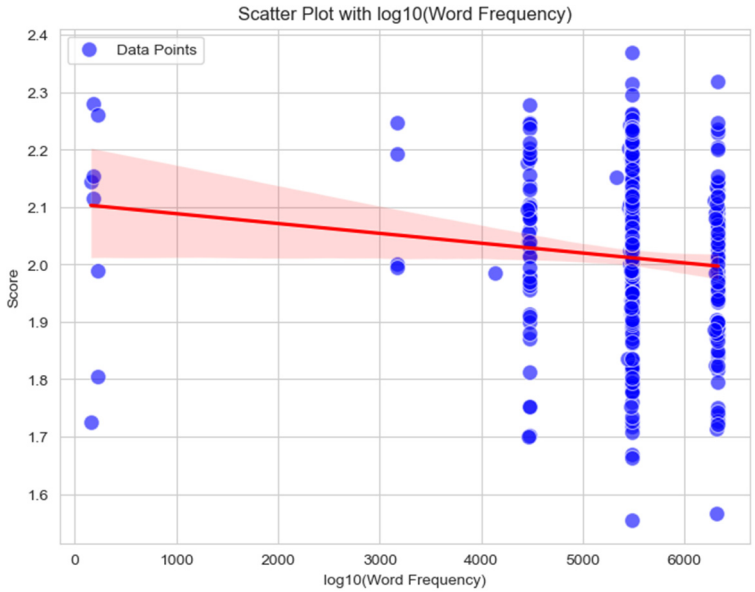

Figure 2. The data regression analysis. The scatter plot illustrates the relationship between the score and the log10 of word frequency, with blue dots representing data samples. The red line indicates the regression trend, and the pink area shows the confidence interval of the regression model, revealing the correlation between word usage frequency and score.

As shown in Figure 3, we analyze the impact of different part-of-speech (POS) tags on the model, with their importance represented by the score. The median scores across categories show notable differences. For example, VBN (verb past participle) has a relatively high median of approximately 2.05, while most other major POS categories, such as RB (adverb), JJ (adjective), and NN (noun singular), cluster around 2.0. The variability of scores also differs across categories. The NN category shows the widest box and whisker range, indicating the most dispersed distribution and highest internal variability, whereas NNS (noun plural) and VBZ (verb third person singular) show narrow boxes, reflecting highly concentrated distributions.

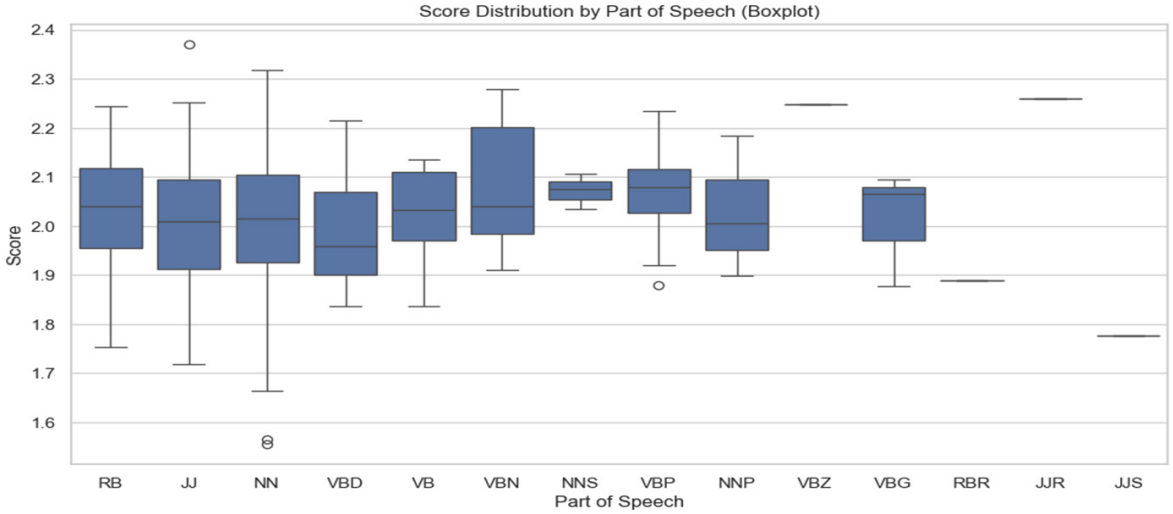

Figure 3. The horizontal axis shows categorical data, listing 13 part-of-speech (POS) tags such as RB (adverb), JJ (adjective), NN (noun singular), and VBN (verb past participle). These tags represent distinct grammatical categories in the dataset. The vertical axis displays a continuous numerical variable called "Score," with values ranging approximately between 1.5 and 2.4. Circles indicate outliers, defined as values more than 1.5 times the interquartile range above the third quartile or below the first quartile.

4.2. The Model Training

In **Wordle**, the action space A is defined as the set of all valid candidate words. If the dictionary contains N valid words, the dimensionality of the action space is $|A| = N$. To enhance strategy efficiency, this study introduces a state-dependent action filtering mechanism, and candidate words are filtered based on the current game state (e.g., revealed letter positions, forbidden letters). In the initial state, the action space corresponds to the full dictionary; in mid-game states (e.g., when the second letter is known to be "A"), the action space is restricted to words where the second letter matches "A".

We use the word frequency, letter frequency, and repetition rate in the dataset as features, and the number of trials as the target variable. To ensure that the input feature and the target variable are in the same scale range, we use the MinMaxScaler to normalize the feature and the target variable to the range $[-1, 1]$. We follow and divide them into a training set and test set and define a custom RegressionEnv reinforcement learning environment. The environment inherits from Gym Env classes. The environment consists of three characteristics: the state space, the movement target space corresponding to the seven variables predicted, and the motion space. The border of the motion space is set to [1, 1], to match the normalized target variable scope. The environment reward function is based on the mean square error (MSE). By selecting a negative MSE as a reward, it encourages the model prediction results to be closer to the true value. The mean action of noise uses smooth periodic updates, set to zero, and the standard deviation is 0.1, in order

to ensure the moderation of exploration. Other hyperparameters of the model include the experience replay buffer size (100,000), the number of learning initial steps (1000), and the training frequency (training every 1000 steps), which are set to balance exploration and exploitation while improving learning efficiency. For the aforementioned hyperparameters, we employ Bayesian optimization to systematically identify the optimal hyperparameter configuration that maximizes policy performances.

The Enhanced-DDPG algorithm, in training, continuously updates the parameters of the critic network and actor network through interaction with the environment. The actor network is responsible for action selection, while the critic network assesses the value of the action. By minimizing the critic network loss function and maximizing the gradient of the actor network strategy, the model prediction ability is optimized step by step.

5. Results

To evaluate the model performance, we perform tests on the test set, restoring the predictions to the original scale by de-normalization. The performance metrics used to evaluate the model include the mean squared error (MSE) and coefficient of determination (R^2). MSE is a measure of the mean squared difference between the predicted value and the true value, while R^2 reflects the model's ability to explain the variation in the data.

In addition, a scatter plot is used to visualize the relationship between the actual and predicted values. The distribution of data points reflects the overall prediction of the model for the data. This method helps us to quickly evaluate the performance of the model, resulting in an MSE = 0.0134, with R^2 = 0.8439 and a scatter plot of predictions as shown in Figure 4.

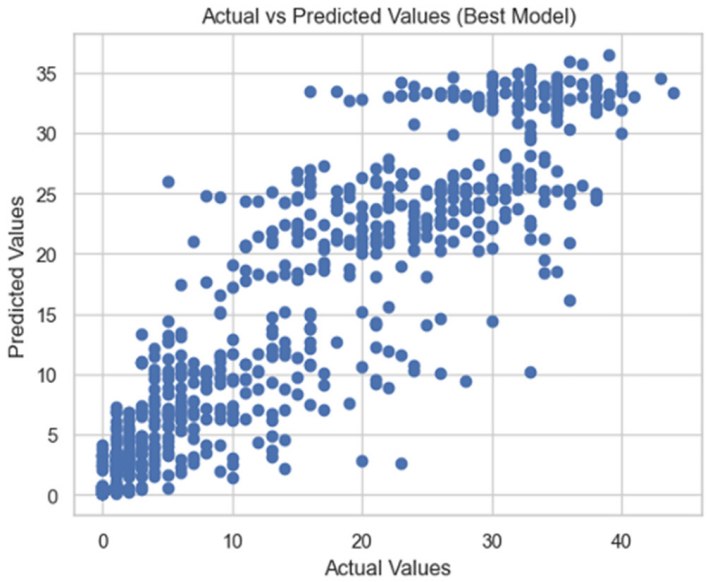

Figure 4. Prediction results of DDPG algorithm. The scatter plot illustrates the comparison between actual values and predicted values predict by the DDPG algorithm, illustrating the distribution and correlation of data points to reflect the algorithm's prediction performance.

5.1. Robustness Analysis

To evaluate the stability of the model in the face of data perturbations and noise, we perform a robustness analysis by adding Gaussian noise ($\sigma = 0.3$) to the original data to observe the change in the prediction performance of the model, as shown in Figure 5. It can be seen that the overall prediction results of the model do not change much after adding noise (represented by red dots).

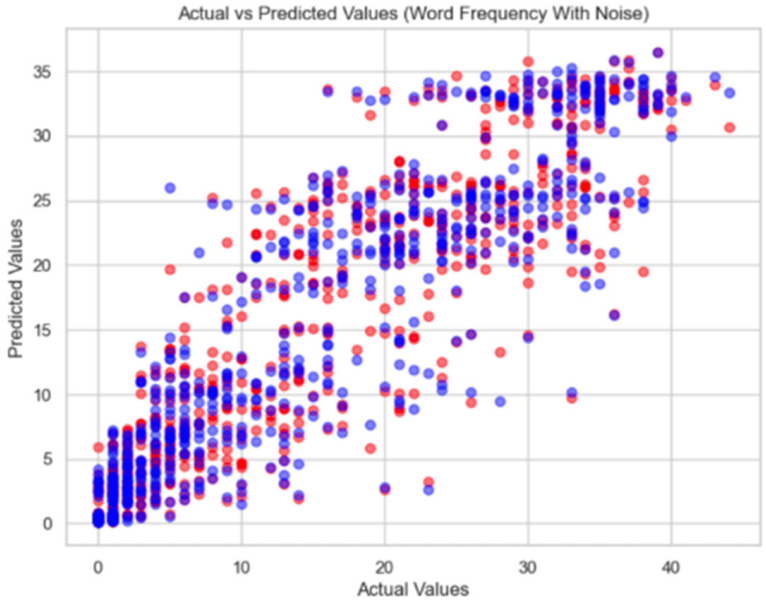

Figure 5. Robustness analysis results of DDPG algorithm. This scatter plot compares actual values with predicted values from DDPG algorithm. Blue dots represent predictions made on the original dataset, and red dots represent the predict result based on Gaussian noise data, showing the algorithm's prediction accuracy and robustness by illustrating the algorithm performance and maintaining stability despite the presence of noise in the data.

5.2. Interpretability Analysis

In the field of deep learning, model interpretability has always been the focus of research on improving model transparency. With the increase in model complexity, how to interpret the prediction results is crucial. As an explanation method based on game theory, SHAP provides a unified explanation framework for the prediction results of deep learning models. The core idea of SHAP is derived from the Shapley value in game theory, which is used to measure the average contribution of a player in a cooperative game. In the displayed shapes, each feature is regarded as a player, and the predictions of a model are the result of the game, calculating the marginal contribution of each feature.

Among all possible feature subsets, the SHAP value is a value assigned to each feature, indicating the contribution of this feature to the prediction result. For this study, the calculation process is as follows:

Step 1: For each sample, generate a sample with a slightly different feature value from the original sample, which is called a perturbed sample.

Step 2: Use the model to predict the prediction results of the original sample and the perturbed sample.

Step 3: Calculate the difference between the prediction results of the perturbed sample and the prediction results of the original sample, which is the SHAP value of the feature.

Step 4: Obtain the SHAP value of each feature by a weighted average of all possible feature subsets.

Finally, we obtain the detailed results, as shown in Figure 6, where the horizontal axis (X-axis) is the SHAP value (i.e., the magnitude and direction of each feature's influence on the model output). The vertical axis (Y-axis) is the feature name (rep, letter frequency, word frequency). The color indicates the size of the feature value, with the color bar on the right. Red indicates a high feature value and blue indicates a low feature value.

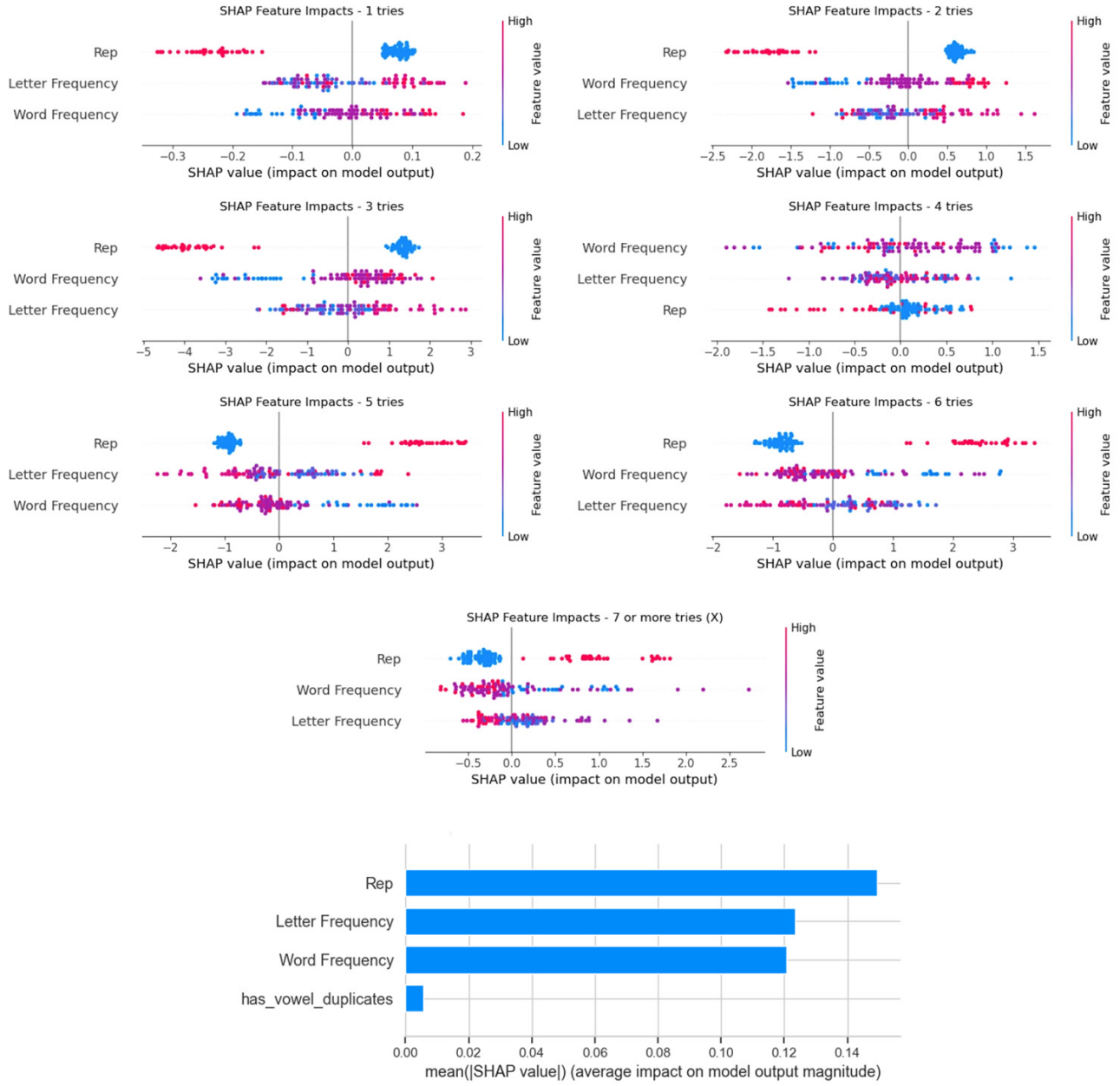

Figure 6. Analysis of SHAP interpretability.

The following is a detail analysis of each subplot within the SHAP summary plot shown in Figure 6.

In the one-try case, rep features display concentrated, positive SHAP values. This suggests that in one-try classification, high-rep features tend to guide the model to predict that class. Letter frequency and word frequency have minimal influence on the model, with SHAP values near zero, indicating that these features have little impact on the judgment of whether it is "one try."

In the two-tries case, the rep continues to have the strongest impact, positively influencing the model output. Word frequency and letter frequency have some negative effects on the model, but these effects are still relatively small. Note that the SHAP values are more widely distributed here, implying that the feature influence is somewhat more complex.

In the three-tries case, word frequency has the largest impact, both positive and negative, with SHAP values ranging from −4 to +3. The SHAP values of letter frequency are more densely clustered around zero but still show some positive and negative effects.

The rep feature also has an effect, though it is slightly weaker than before, with less scatter concentration.

In the four-tries case, the SHAP values for all three features are relatively balanced, indicating that the model relies on multiple features when judging "four tries." The lack of dominant features suggests that this class is more complex or more evenly distributed in the training data.

In the five-tries case, the SHAP values for rep are highly concentrated but negative in direction, indicating that low values push down the model output. letter frequency and word frequency become important here. word frequency, in particular, has numerous high values scattered in the region of positive SHAP values, suggesting that this class may be judged by multiple dimensions.

In the six-tries case, negative SHAP values for rep are still evident, with high values (red) driving predictions down. The influence range of letter frequency is expanded, with the positive driving effect increased. Meanwhile, the distribution of word frequency is scattered, showing a certain positive influence.

For the case involving seven or more tries (X), the impacts of rep features remain dominant, with SHAP values concentrated and negative. Both word frequency and letter frequency exhibit positive effects, with high values more likely to prompt the model to classify as this class. The model associates letter/word frequency with frequent attempts.

In addition, we analyze the impact of vowel repetition patterns on the model. As shown in the horizontal bar chart of Figure 6, the presence or absence of vowel repetition patterns has a minimal impact on the model's results.

Finally, the rep feature has a strong influence on the model output in most sub-graphs. When the number of trials is small, its influence is positive (that is, a high value promotes the prediction), and when the number of trials is large, its influence is negative (that is, a high value inhibits the prediction).

Letter frequency and word frequency become more important in the medium and high number of trials (3–7+), especially in the "three" and "five" trials.

In terms of color (i.e., feature value), words or letters with high frequencies tend to increase the model's likelihood of predicting "more attempts," while low frequencies do the opposite.

5.3. Model Comparison

To further evaluate the performance of Enhanced-DDPG algorithm, we use the Enhanced-DDPG algorithm to conduct comparison experiments with RFR, XGBoost, METRA, SQIRL, and LightGBM. The hyperparameter settings of RFR, XGBoost, METRA, SQIRL, and LightGBM are shown in Table 1.

We train and evaluate the above model on the original dataset and the dataset with Gaussian noise, and we use the two indicators of MSE and R^2 to compare the results, as shown in Table 2.

On the original dataset, Enhanced-DDPG has an MSE of 0.0134 and an R^2 of 0.8439, which are significantly better values than those for RFR (MSE = 0.0425, R^2 = 0.6554), XGBoost (MSE = 0.0276, R^2 = 0.7960), LightGBM (MSE = 0.0221, R^2 = 0.8342), METRA (MSE = 0.0242, R^2 = 0.6977), and SQIRL (MSE = 0.0371, R^2 = 0.7153). This reflects that Enhanced-DDPG has better performance in processing raw data. This is due to the dynamic learning mechanism of Enhanced-DDPG, which makes it better able to capture nonlinear and complex relationships in the data.

Table 1. Hyperparameter settings of the models, including RFR, XGBoost, LightGBM, METRA, and SQIRL.

	Hyperparameters	**Value**
RFR	max_depth	10
	min_samples_leaf	4
	min_samples_split	10
	n_estimators	100
XGBoost	max_depth	5
	learning_rate	0.01
	n_estimators	100
	subsample	0.5
	colsample_bytree	0.5
LightGBM	max_depth	3
	learning_rate	0.01
	n_estimators	100
	num_leaves	20
METRA	learning_rate	3×10^{-4}
	discount_factor	0.99
	latent_dimension	32
	random_exploration-steps	-
SQIRL	learning_rate	10^{-3}
	discount_factor	0.95
	latent_dimension	-
	random_exploration-steps	10,000

Table 2. Performance comparison of different models, including Enhanced-DDPG, RFR, XGBoost, LightGBM, METRA, and SQIRL, on raw and Gaussian noise datasets, using MSE and R^2 metrics.

Model	**MSE**	**R^2**
Enhanced-DDPG (Raw data)	0.0134	0.8439
RFR (Raw data)	0.0425	0.6554
Enhanced-DDPG (Gaussian noise data)	0.0241	0.7853
RFR (Gaussian noise data)	0.0512	0.6133
XGBoost (Raw data)	0.0276	0.7960
LightGBM (Raw data)	0.0221	0.8342
XGBoost (Gaussian noise data)	0.0314	0.7215
LightGBM (Gaussian noise data)	0.0325	0.7749
METRA (Raw data)	0.0242	0.6977
SQIRL (Raw data)	0.0371	0.7153
METRA (Gaussian noise data)	0.0211	0.6991
SQIRL (Gaussian noise data)	0.0281	0.7186

On the dataset with Gaussian noise, Enhanced-DDPG has an MSE = 0.0241 and R^2 = 0.7853, which is still better than RFR (MSE = 0.0512, R^2 = 0.6133), XGBoost (MSE = 0.0314, R^2 = 0.7215), LightGBM (MSE = 0.0325, R^2 = 0.7215), METRA (MSE = 0.0211, R^2 = 0.6991), and SQIRL (MSE = 0.0281, R^2 = 0.7186). The introduction of noise in the dataset leads to the performance degradation of Enhanced-DDPG algorithm, but the decline is

significantly smaller than that of traditional methods, indicating that Enhanced-DDPG has stronger robustness in noisy environments and can better adapt to fluctuations in data.

Ablation experiments compare the original DDPG (without the SHAP module) and DDPG with representative features removed against our proposed Enhanced-DDPG. As shown in Table 3, when using the original reward function instead of the SHAP-based reward and removing the highest-contribution features, the algorithm's performance degrades significantly. This demonstrates the low redundancy and high reliability of Enhanced-DDPG and confirms that our SHAP-based reward function can efficiently and accurately guide the model's training process.

Table 3. Results of ablation experiments comparing the proposed Enhanced-DDPG, DDPG without representative (rep) features, and the original DDPG algorithm in terms of MSE and R^2 performance metrics.

Model	MSE	R^2
DDPG (Our Enhanced-DDPG)	0.0134	0.8439
DDPG (Without rep features)	0.0517	0.5230
DDPG (Original algorithm)	0.0298	0.6821

Based on the above comparison and analysis, it is easy to see that Enhanced-DDPG performs better than other models on both raw data and noisy data. The results show that Enhanced-DDPG has obvious advantages in dealing with dynamic and complex problems, while RFR is more suitable for dealing with structured and stable data. In addition, the robustness of Enhanced-DDPG enables it to maintain good performance in noisy environments, which provides support for its wide application in practical scenarios.

6. Conclusions

By integrating NLP, I-AI, and RL techniques, this study proposes a transparent, efficient, and robust AI system for text prediction (using **Wordle** as an example). Specifically, the reinforcement learning method based on the DDPG algorithm shows significant advantages in the task of predicting the word-trial number distribution of **Wordle**, and its mean square error (MSE = 0.0134) and coefficient of determination (R^2 = 0.8439) are better than the traditional RFR, XGBoost, and RL methods. By introducing the SHAP interpretability analysis tool, we reveal the positive impact of the feature "repeated Letter Pattern (rep)" in the model's decision logic on the prediction of a low number of attempts, as well as the key role of the features "word frequency" and "letter frequency" in the scenario with a high number of attempts. The robustness analysis of the model shows that Enhanced-DDPG can still maintain high prediction performance in datasets with Gaussian noise (MSE = 0.0241, R^2 = 0.7853), which verifies its stability in dynamic and complex scenes.

While its application in natural language processing (NLP) is still an emerging area, DDPG holds promise for tasks that can be framed with a continuous-action space. For instance, in dialogue systems, DDPG could potentially be used to fine-tune the emotional tone or style of a generated response on a continuous spectrum. In text generation, it could be used to control various attributes of the generated text, such as its level of formality or complexity, in a continuous manner. These applications are still in the exploratory phase but represent a new frontier for DDPG and reinforcement learning in the realm of language.

Finally, although the model proposed in this study demonstrates high performance and excellent robustness in predicting **Wordle** text data, there is still room for improvement. The data scale and feature types processed by the current model are limited. We focus on processing multi-domain and cross-language datasets and further integrate adaptive feature

selection and multi-modal data fusion techniques to further improve the generalization and decision-making ability of the model in practical applications. In addition, more efficient explanatory tools are combined to realize the transparency of multi-angle models, and an efficient and easy-to-understand prediction model is constructed. We believe that these improvements can help to apply this research to a wider range of complex dynamic decision-making scenarios and promote the development of related theories and technologies.

Author Contributions: J.Z. and J.J. participated in the data collection and manuscript drafting. S.W. and R.Y. participated in the data collection. L.Y. participated in the data pre-processing. J.Z., L.Z. and L.Y. participated in the design and statistical analysis of the study. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Anhui Provincial Department of Education's Higher Education Institution Scientific Research Project (Humanities and Social Sciences) in Anhui Province in 2020, grant number SK2020A0373; Quality Engineering Project in Anhui Province in 2023, grant number 2023dzxkc030; Higher Education Institution Scientific Research Project (Humanities and Social Sciences) in Anhui Province in 2020, grant number 2024AH040383; Student Innovation Training Program in Anhui Province in 2024, grant number 202410368070.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author(s).

Acknowledgments: We sincerely express our gratitude to all participants.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
2. Biau, G.; Scornet, E. A random forest guided tour. *Test* **2016**, *25*, 197–227. [CrossRef]
3. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
4. Bentejac, C.; Csorgő, A.; Martinez-Munoz, G. A comparative analysis of gradient boosting algorithms. *Artif. Intell. Rev.* **2021**, *54*, 1937–1967. [CrossRef]
5. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*; Neural Information Processing Systems Foundation, Inc. (NeurIPS): San Diego, CA, USA, 2017.
6. Li, S.; Dong, X.; Ma, D.; Dang, B.; Zang, H.; Gong, Y. Utilizing the lightgbm algorithm for operator user credit assessment research. *arXiv* **2024**, arXiv:2403.14483. [CrossRef]
7. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
8. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1587–1596.
9. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In Proceedings of the International Conference on Machine Learning, PMLR, Beijing, China, 22–24 June 2014; pp. 387–395.
10. Balduzzi, D.; Ghifary, M. Compatible value gradients for reinforcement learning of continuous deep policies. *arXiv* **2015**, arXiv:1509.03005.
11. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft Actor-Critic Algorithms and Applications. *arXiv* **2019**, arXiv:1812.05905. [CrossRef]
12. Park, S.; Rybkin, O.; Levine, S. METRA: Scalable Unsupervised RL with Metric-Aware Abstraction. *arXiv* **2024**, arXiv:2310.08887. [CrossRef]
13. Laidlaw, C.; Zhu, B.; Russell, S.; Dragan, A. The Effective Horizon Explains Deep RL Performance in Stochastic Environments. *arXiv* **2024**, arXiv:2312.08369. [CrossRef]
14. Reddi, A.; Tölle, M.; Peters, J.; Chalvatzaki, G.; D'Eramo, C. Robust Adversarial Reinforcement Learning via Bounded Rationality Curricula. *arXiv* **2023**, arXiv:2311.01642. [CrossRef]

15. Futuhi, E.; Karimi, S.; Gao, C.; Müller, M. ETGL-DDPG: A Deep Deterministic Policy Gradient Algorithm for Sparse Reward Continuous Control. *arXiv* **2025**, arXiv:2410.05225. [CrossRef]
16. Hickling, T.; Zenati, A.; Aouf, N.; Spencer, P. Explainability in deep reinforcement learning: A review into current methods and applications. *ACM Comput. Surv.* **2023**, *56*, 1–35. [CrossRef]
17. Anderson, A.; Dodge, J.; Sadarangani, A.; Juozapaitis, Z.; Newman, E.; Irvine, J.; Chattopadhyay, S.; Fern, A.; Burnett, M. Explaining Reinforcement Learning to Mere Mortals: An Empirical Study. *arXiv* **2019**, arXiv:1903.09708. [CrossRef]
18. Wang, C.; Wu, L.; Yan, C.; Wang, Z.; Long, H.; Yu, C. Coactive design of explainable agent-based task planning and deep reinforcement learning for human-UAVs teamwork. *Chin. J. Aeronaut.* **2020**, *33*, 2930–2945. [CrossRef]
19. Douglas, N.; Yim, D.; Kartal, B.; Hernandez-Leal, P.; Maurer, F.; Taylor, M.E. Towers of saliency: A reinforcement learning visualization using immersive environments. In Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces, Daejeon, Republic of Korea, 10–13 November 2019; pp. 339–342.
20. Huber, T.; Schiller, D.; Andre, E. Enhancing explainability of deep reinforcement learning through selective layer-wise relevance propagation. In *KI 2019: Advances in Artificial Intelligence: 42nd German Conference on AI, Kassel, Germany, September 23–26, 2019*; Proceedings 42; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 188–202.
21. Muschalik, M.; Baniecki, H.; Fumagalli, F.; Kolpaczki, P.; Hammer, B.; Hullermeier, E. shapiq: Shapley interactions for machine learning. In *Advances in Neural Information Processing Systems 37*; Neural Information Processing Systems Foundation, Inc. (NeurIPS): San Diego, CA, USA, 2024; pp. 130324–130357.
22. Aas, K.; Jullum, M.; Løland, A. Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. *Artif. Intell.* **2021**, *298*, 103502. [CrossRef]
23. Baniecki, H.; Kretowicz, W.; Piątyszek, P.; Wiśniewski, J.; Biecek, P. Dalex: Responsible machine learning with interactive explainability and fairness in python. *J. Mach. Learn. Res.* **2021**, *22*, 1–7.
24. Oksuz, A.C.; Halimi, A.; Ayday, E. AUTOLYCUS: Exploiting Explainable Artificial Intelligence (XAI) for Model Extraction Attacks against Interpretable Models. In Proceedings of the Privacy Enhancing Technologies, Bristol, UK, 15–20 July 2024.
25. Ribeiro, M.T.; Singh, S.; Guestrin, C. Model-Agnostic Interpretability of Machine Learning. *arXiv* **2016**, arXiv:1606.05386. [CrossRef]
26. Lundberg, S.M.; Erion, G.; Chen, H.; DeGrave, A.; Prutkin, J.M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; Lee, S.-I. From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.* **2020**, *2*, 56–67. [CrossRef]
27. Bala, R.; Sharma, A.; Goel, N. Comparative analysis of diabetic retinopathy classification approaches using machine learning and deep learning techniques. *Arch. Comput. Methods Eng.* **2024**, *31*, 919–955. [CrossRef]
28. Letoffe, O.; Huang, X.; Marques-Silva, J. On correcting SHAP scores. *arXiv* **2024**, arXiv:2405.00076v1.
29. Choi, J.; Kim, H.; Noh, T.; Kang, Y.J.; Noh, Y. Digital Twin Data-Driven Multi-Disciplinary and Multi-Objective Optimization Framework for Automatic Design of Negative Stiffness Honeycomb. *Int. J. Precis. Eng. Manuf.* **2023**, *24*, 1453–1472. [CrossRef]
30. Feretzakis, G.; Sakagianni, A.; Anastasiou, A.; Kapogianni, I.; Bazakidou, E.; Koufopoulos, P.; Koumpouros, Y.; Koufopoulou, C.; Kaldis, V.; Verykios, V.S. Integrating Shapley values into machine learning techniques for enhanced predictions of hospital admissions. *Appl. Sci.* **2024**, *14*, 5925. [CrossRef]
31. Salih, A.; Raisi-Estabragh, Z.; Galazzo, I.B.; Radeva, P.; Petersen, E.; Menegaz, G.; Lekadir, K. Commentary on explainable artificial intelligence methods: SHAP and LIME. *arXiv* **2023**, arXiv:2305.02012.
32. Yang, G.; Ye, Q.; Xia, J. Unbox the black-box for the medical explainable AI via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond. *Inf. Fusion* **2022**, *77*, 29–52. [CrossRef]
33. Baehrens, D.; Schroeter, T.; Harmeling, S.; Kawanabe, M.; Hansen, K.; Muller, K.R. How to explain individual classification decisions. *J. Mach. Learn. Res.* **2010**, *11*, 1803–1831.
34. Adebayo, J.; Gilmer, J.; Muelly, M.; Goodfellow, I.; Hardt, M.; Kim, B. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems 31*; Neural Information Processing Systems Foundation, Inc. (NeurIPS): San Diego, CA, USA, 2018.
35. Ribeiro, M.T.; Singh, S.; Guestrin, C. Why should I trust you? Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144.
36. Teso, S.; Alkan, O.; Daly, E.; Stammer, W. Explanations in Interactive Machine Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 22 February–1 March 2022.