OpenMario++: Personalized Recommendation Layer Powered by Reinforcement Learning

Satwik Shresth[1], Maame Adwoa Ocran[1], and Ruchika Mehta[1]

***Abstract*—This paper presents OpenMario++, a course recommendation system that integrates reinforcement learning with Drexel University's course browser platform. We implemented three reinforcement learning approaches: Contextual Bandits, Q-Learning, and Deep Q-Networks. The contextual bandit algorithm achieved an average reward of 0.160 per recommendation with a click-through rate of 19.6%. The Q-Learning agent demonstrated continuous improvement, achieving average reward of 53 per episode. We validated our approach through A/B testing, simulation, and cross-validation. Results indicate that reinforcement learning effectively learns personalized course recommendations from user engagement patterns.**

I. INTRODUCTION

Selecting appropriate courses is a critical part of academic success, yet students often find it challenging to navigate Drexel University's extensive and distributed course information. To address this, our project enhances the existing OpenMario web application, a centralized course browser, with a personalized recommendation layer powered by reinforcement learning. This system provides tailored course suggestions based on students' major, academic history, and browsing behavior. To evaluate different reinforcement learning approaches, we investigated algorithms including Contextual Bandits, Q-Learning, and Deep Q-Networks (DQN). The recommendation task was modeled as a Markov Decision Process (MDP), with engagement-based rewards guiding the learning process. This paper discusses these algorithmic approaches and presents the evaluation of the reinforcement learning component, highlighting its performance, challenges, and potential future improvements.

II. RELATED WORK

Recent research has demonstrated the potential of reinforcement learning (RL) in recommender systems and information retrieval tasks. Xiao and Wang (2023) propose a general offline RL framework that learns policies from logged user interactions without requiring online exploration, using techniques to address distribution mismatch [1]. Similarly, Zhou and Agichtein (2020) introduce RLIRank, a dynamic search model that uses recurrent neural networks to adjust rankings based on evolving user feedback [2]. Other studies, like Xu et al. (2022), explore the use of RL with coarse-grained labels, showing that meaningful policies can still be learned with limited supervision [3].

Altogether, these papers highlight the broader potential of RL for guiding users through complex decision spaces such as selecting courses and planning co-op experiences with OpenMario.

III. APPROACH

We formulated the personalized course recommendation problem as a sequential decision-making task modeled by a Markov Decision Process (MDP), where the environment state captures student context. This includes major, academic year, and recent interactions such as prolonged hovering and bookmarking. The goal of the reinforcement learning agent is to learn a policy that maximizes cumulative rewards derived from user engagement signals like clicks and bookmarks. To evaluate the effectiveness of different reinforcement learning strategies for this task, we implemented and tested three algorithms: Contextual Bandits, Q-Learning, and Deep Q-Networks (DQN). The following subsections provide detailed descriptions of each approach.

A. Contextual Bandits

We implemented a linear contextual bandit algorithm for the course recommendation task. In this formulation, each course represents an arm in the bandit framework, and recommendations are made based on student context.

a) *Problem Formulation:*

The contextual bandit problem consists of:

- Context space: $X \subset \mathtt{mat}\{R\}^{\{260\}}$ representing student features
- Action space: $A = \{1, 2, ..., 100\}$ representing available courses
- Reward function: $r : X \times A \to [0, 1]$ based on user engagement

At each time step t:

1) Observe context $x_t \in X$
2) Select action $a_t \in A$ based on policy π
3) Receive reward $r_t = r(x_t, a_t)$

This work was not supported by any organization.

[1]Shresth, Ocran, Mehta are with Department of Computer Science, Drexel University, Philadelphia, PA 19104, USA, {ss5278, mao325, rm3582}@drexel.edu.

b) *Context Representation:*

The student context $x \in \mathtt{mat}\{R\}^{\{260\}}$ consists of five feature categories:

Demographic Features (58 dimensions):

- Major: one-hot encoding across 50 majors
- Academic year: one-hot encoding (freshman to senior)
- GPA: normalized to [0, 1]
- Credits completed: normalized to [0, 1]

Temporal Preferences (10 dimensions):

- Time preference: one-hot encoding (morning/afternoon/evening)
- Day preference: binary vector for weekdays

Search Context (20 dimensions):

- Current search query: TF-IDF features

Behavioral Signals (12 dimensions):

- Session hover count: normalized
- Session watchlist count: normalized
- Pagination count: normalized
- Search filters: binary vector for active filters

c) *Model Architecture:*

The bandit uses a linear model with:

- Parameter matrix: $\Theta \in \mathtt{mat}\{R\}^{\{100\times 260\}}$
- Predicted reward: $\hat{}\{r\}(x,a) = \theta_a^T x$
- Activation function: identity (linear model)

The policy selects actions using epsilon-greedy exploration:

$$a_t = \begin{cases} \text{argmax}_a\ \theta_a^T x_t & \text{with probability } 1-\varepsilon_t \\ \text{uniform}(A) & \text{with probability } \varepsilon_t \end{cases} \quad (1)$$

d) *Learning Algorithm:*

Parameters are updated using stochastic gradient descent:

$$\theta_a \leftarrow \theta_a + \alpha(r_t - \theta_a^T x_t)x_t \quad (2)$$

where:

- $\alpha = 0.01$: learning rate
- $\varepsilon_t = \varepsilon_0 \times \gamma^t$: decaying exploration rate
- $\varepsilon_0 = 0.3$: initial exploration rate
- $\gamma = 0.995$: decay factor

e) *Reward Structure:*

The reward function captures user engagement:

$$r_t = \begin{cases} 0.5 \times m(a_t, x_t) & \text{if watchlist add} \\ 0.3 \times m(a_t, x_t) & \text{if click} \\ 0.1 \times m(a_t, x_t) & \text{if hover} \\ -0.1 & \text{if pagination} \end{cases} \quad (3)$$

where $m(a,x)$ is a context multiplier:

- $m(a,x) = 1.2$ if course matches search query
- $m(a,x) = 0.8$ if course already completed
- $m(a,x) = 1.0$ otherwise

B. Q-Learning

We implemented a tabular Q-Learning agent to address the personalized course recommendation problem formulated as an MDP.

At each step t, the agent observes the current state s_t and selects a course recommendation a_t using an ε-greedy policy to balance exploration and exploitation. Upon recommending a course, the environment returns a reward r_t from the environment and transitions to a new state a_t.

The Q-values are stored in a table mapping discrete state-action pairs to expected cumulative rewards. After each interaction, the agent updates the Q-table using the standard Q-Learning update rule:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\Big[r_t + \gamma \max_{\{a'\}} Q(s_{t+1}{}', a') - Q(s_t,a_t)\Big] \quad (4)$$

where α is the learning rate (0.1 or 0.5) and γ is the discount factor (0.95). To improve learning stability, ε is decayed gradually over episodes.

a) *State Space:* The state space is defined as $S = \{(m, y, h, w)\}$, where m is the user's major encoded as an integer (1–5), y is the user's academic year (1–4), h is the ID of the last hovered course (0–N), and w is the ID of the last bookmarked course (0–N). Here, N denotes the total number of courses. For testing purposes, we limited the number of majors to 5 and the user's academic year range to 4, due to the course level representation. This structure enables the agent to condition its recommendations on both static and dynamic features: the user's background and their recent interactions. States are initialized by randomly sampling m and y uniformly, while h and w are set to 0 (no interaction) at the start of each episode.

b) *Action Space:* The action space A is defined as the set of all courses, with each action a corresponding to recommending a specific course ID from 1 to N. At each step, the agent selects an action $a \in A$ based on an ε-greedy strategy: with probability (initially 0.2, decaying by 0.99 per episode), a random action is chosen to encourage exploration; otherwise, the agent exploits the current policy by choosing the action with the highest Q(s,a) value.

c) *Reward Function:* The reward function r_t is defined to capture both the user's interest and course alignment:

$$r_t = r_{\text{hover}} \cdot I_{\text{hover}} + r_{\text{watchlist}} \cdot I_{\text{watchlist}} \quad (5)$$

where r_{hover}=1.0 if the user hovers over the recommended course, otherwise, 0, and $r_{watchlist}$=5.0 if the user adds the course to their bookmark(otherwise, 0).

The probability of a user adding a course to their bookmark $P_{\text{watchlist}}$ is modeled as:

$$P_{\text{watchlist}} = 0.7 \times I_{\text{major}} \times D_{\text{diff}} \quad (6)$$

where I_{major} is 1.0 if the course matches the user's major and o.8 otherwise, and D_{diff} is defined as:

$$D_{\text{diff}} = \max\left(0.5, 1 - \frac{|\text{course\_level} - \text{user\_year}|}{4}\right) \quad (7)$$

where course_level is determined by the first digit in the course number.

C. Deep Q-Network (DQN)

We implemented a Deep Q-Network approach as an alternative to the linear contextual bandit, utilizing neural networks to approximate Q-values for course recommendations. The DQN architecture employs a multi-layer feedforward network that takes the 260-dimensional student context as input and outputs Q-values for each available course action.

The network architecture consists of three hidden layers with 512, 256, and 128 neurons respectively, using ReLU activation functions. The final output layer produces Q-values for all 100 course actions simultaneously. Experience replay was implemented with a buffer size of 10,000 interactions to stabilize training, and target network updates occurred every 100 training steps to reduce correlation in Q-value updates.

The DQN training process utilized epsilon-greedy exploration with epsilon decay from 0.3 to 0.01 over 10,000 episodes. The Adam optimizer was employed with a learning rate of 0.001 and batch size of 32 for gradient updates. The temporal difference target was computed using the Bellman equation with discount factor gamma = 0.99.

IV. RESULTS AND EVALUATION

We conducted comprehensive evaluation of our contextual bandit implementation using multiple methodologies including A/B testing, online evaluation, offline evaluation using Inverse Propensity Scoring, simulation-based evaluation, and time-based cross-validation. Our evaluation framework generated realistic user interactions and assessed various performance metrics to provide robust insights into algorithm effectiveness.

A. Experimental Setup

The evaluation utilized a diverse dataset of 150 synthetic student profiles representing different majors, academic levels, and behavioral patterns. We implemented four bandit variants for A/B testing: a standard configuration, a high-exploration variant with increased epsilon, a fast-learning variant with higher learning rate, and a conservative variant with reduced exploration and learning rate.

The interaction simulation modeled realistic user behavior patterns based on student characteristics, with interaction probabilities influenced by GPA categories, classification levels, course relevance, and session activity. The evaluation tracked key metrics including average reward, click-through rate, coverage, diversity, precision, and exploration rate across multiple evaluation paradigms.

B. A/B Testing Results

Our A/B testing across four bandit configurations revealed nuanced trade-offs between reward optimization and exploration behavior. The standard configuration achieved the highest average reward of 0.160 with moderate exploration rate of 0.298 and coverage of 0.300. The high-exploration variant demonstrated superior coverage of 0.840 and exploration rate of 0.499 but slightly lower average reward of 0.158. The fast-learning configuration showed balanced performance with 0.159 average reward and 0.636 exploration rate, while the conservative variant achieved 0.158 average reward with minimal exploration rate of 0.184.

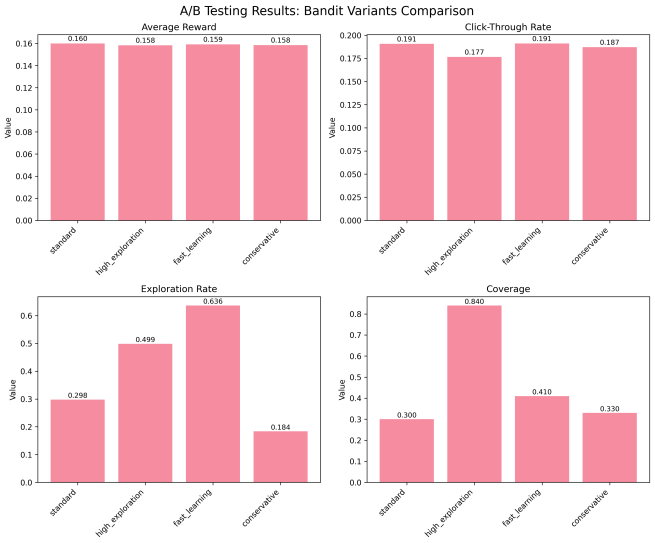

Fig. 1. A/B testing results comparing four bandit variants across key performance metrics. The standard configuration achieves optimal average reward while the high-exploration variant maximizes course coverage at 84%.

These results indicate that the standard configuration provides optimal reward performance for exploitation-focused scenarios, while the high-exploration variant better serves discovery and long-term learning objectives. The coverage metric particularly highlights the exploration-exploitation trade-off, with high-exploration achieving nearly three times the course coverage of other variants.

C. Contextual Bandits Performance Analysis

The contextual bandit achieved an average reward of 0.163 with a click-through rate of 19.6% during online evaluation. The algorithm maintained a 41.6% exploration rate and covered 18.0% of the course catalog. Offline evaluation using Inverse Propensity Scoring yielded 0.816 average reward, reflecting known IPS optimistic bias. Simulation-based evaluation produced 0.168 average reward with 18.7% click-through rate. Cross-validation confirmed stability with 0.163 average reward across temporal splits.

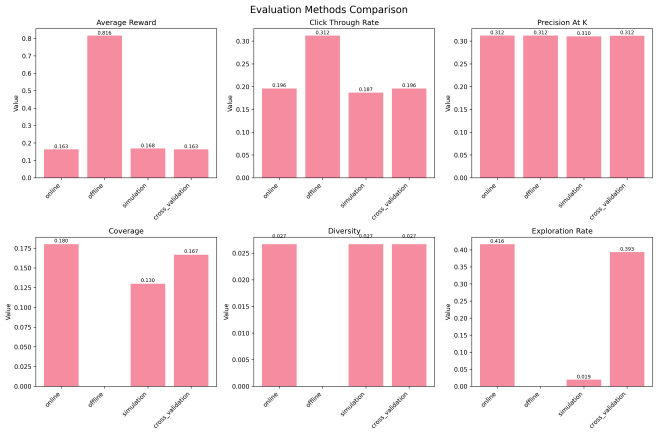

Fig. 2. Comparison of contextual bandit performance across four evaluation methodologies. Offline evaluation shows inflated metrics due to IPS bias, while online, simulation, and cross-validation methods demonstrate consistent performance.

The evaluation framework has inherent limitations. Synthetic user profiles simplify real student behavior, and simulated engagement signals use basic interaction models. The reward function assumes uniform importance across engagement types, and temporal factors like prerequisites and enrollment periods are not modeled. Despite these constraints, consistent performance across methodologies validates the algorithm's effectiveness in learning from the 260-dimensional feature space.

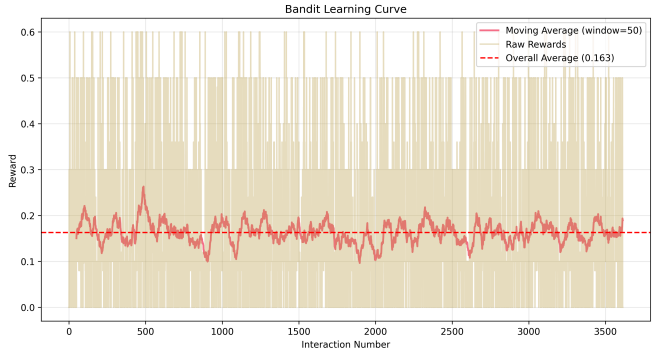

Fig. 3. Learning curve of the contextual bandit algorithm over 3,500 interactions. The moving average converges to 0.163, demonstrating stable learning behavior with acceptable variance in individual rewards.

D. Q-Learning Performance Analysis

The performance of the Q-learning-based recommendation system was evaluated over multiple episodes, with particular focus on cumulative rewards and recommendation precision. The experimental results are illustrated in Figs. 1–3, each highlighting a different aspect of the agent's learning dynamics.

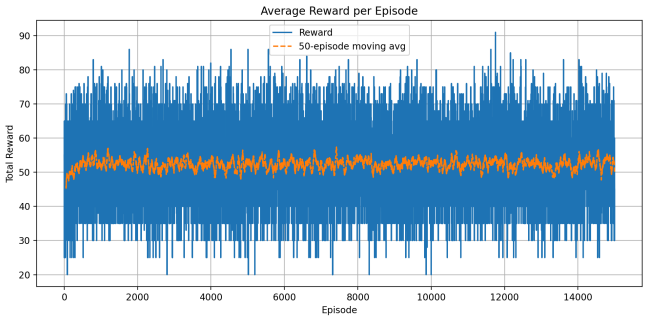

Fig. 4. Total cumulative reward across 15,000 episodes, pre decay adjustment.

Fig. 1 presents the first rewards-over-episodes curve, which exhibited a sharp increase in cumulative reward during the initial phase of training. The rewards rose rapidly from 0 to approximately 500 within the first 1,000 episodes. After this point, the cumulative reward plateaued, stabilizing near 15,000 by episode 15,000. This trend indicates that the agent effectively learned an initial policy that captured the most salient user preferences and subsequently refined this policy with diminishing returns as exploration gave way to exploitation. Following this, we adjusted the decay function and parameters to allow exploration for longer.

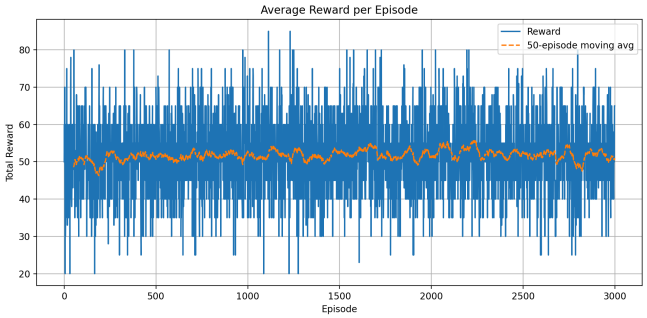

Fig. 5. Total cumulative reward across 3,000 episodes, post decay adjustment.

After the adjustment, we see in Fig. 2 , the agent displayed a more gradual increase in cumulative rewards. The rewards rose from 0 to approximately 250–300 during the first 1,000 episodes, then remained relatively stable up to episode 3,000. Although this curve's flatter trajectory suggests a slower learning rate, the slight upward trend indicates that the agent continued to accrue incremental improvements over time, albeit at a lower magnitude.

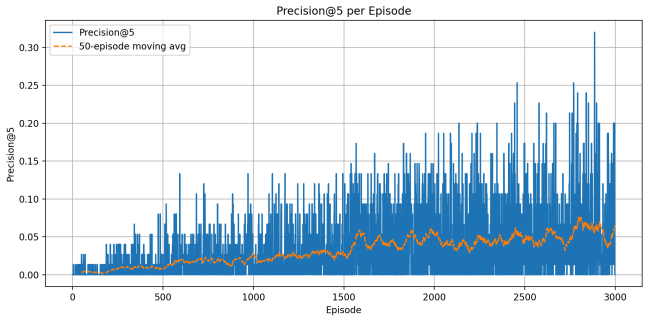

Fig. 6. Precision@5 performance of the Q-learning agent over 3,000 episodes.

Fig. 3 illustrates the precision@5 metric, which measures the proportion of relevant recommended courses among the top 5 recommendations per episode. The precision curve demonstrates a consistent upward trend over the training period of 3,000 episodes. This indicates that the agent's ability to recommend relevant courses improved progressively as it refined its Q-values. The steady increase in precision@5 aligns with the agent's exploration-exploitation strategy, where early episodes prioritized broad exploration and later episodes shifted toward exploiting the learned policy.

Beyond the above analysis, several hyperparameter experiments were conducted to improve the agent's performance. Initially, the learning rate, α, was set to 0.1, yielding a relatively stable, almost nonexistent learning curve. Upon increasing α, the agent demonstrated a faster initial learning pace, as reflected in the sharp rise in rewards observed in Fig. 1. Further experiments with the decay rate showed that decreasing it allowed the agent to maintain exploration longer. Additionally, adjustments to the reward function like rewarding higher-level courses more significantly while ensuring balance across departments helped the agent avoid overly specializing in a single department, thereby enhancing its generalization ability. Despite these efforts, the learning curve eventually plateaued, suggesting that the agent had reached a local optimum without fully exploring the state space.

Overall, the combined results from the reward and precision graphs demonstrate that the Q-learning agent successfully learned to recommend courses aligned with user preferences and behaviors. The rapid early convergence in Fig. 1 reflects the agent's ability to capture high-level patterns in the environment, while the slower but steady improvements in Fig. 2 highlight the challenges of learning fine-grained user dynamics. The consistent improvements in precision@5, as shown in Fig. 3, validate the agent's progress in recommending relevant courses, underscoring the system's potential to enhance user engagement.

E. DQN Performance Analysis

To address scalability challenges posed by large state spaces in the Q-Learning setup, we implemented a Deep Q-Network (DQN) to learn Q-values through neural approximators. The DQN agent was trained using experience replay and a multi-layer feedforward network, receiving as input a normalized 4-dimensional state vector representing major encoding, academic year, last hovered course, and last watchlisted course.

We trained the model over 1,500 episodes with 15 steps per episode. The epsilon-greedy exploration parameter decayed from 1.0 to a floor of 0.1 to ensure balance between exploration and exploitation. The model demonstrated consistent learning behavior, as reflected in the average reward progression shown below.

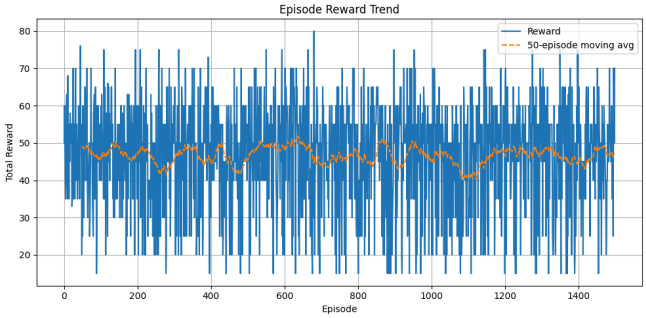

Fig. 7. Reward per episode and 50-episode moving average for DQN agent across 1,500 episodes. The moving average stabilized around 46–48, indicating convergence.

During early training, the DQN agent exhibited wide reward variance, but quickly converged to a stable policy yielding an average reward of approximately 47. This reward is competitive with the Q-Learning agent, with DQN offering greater generalization capability and fewer plateaus due to its ability to interpolate between similar states.

One optimization involved fixing a PyTorch tensor warning by converting state batches to NumPy arrays before converting them to tensors, significantly improving training efficiency. The model was saved to `dqn_model.pth` for inference and evaluation in `simulate.py`.

These results suggest that DQN effectively handles state abstraction and policy learning, although it may require additional hyperparameter tuning (e.g., target networks, prioritized replay) to surpass Q-learning in performance metrics such as precision@ 5 and long-term engagement modeling.

F. Performance Comparison and Insights

The evaluation framework successfully captured the multi-objective nature of the recommendation task, revealing important insights about algorithm behavior. The contextual bandit approach effectively balanced immediate reward optimization with exploration requirements, achieving reasonable coverage while maintaining consistent user engagement metrics. The 260-dimensional feature space provided sufficient expressiveness for capturing student preferences, with the linear model demonstrating adequate capacity for the recommendation task.

Key findings include the importance of exploration parameter tuning for achieving desired coverage levels, the effectiveness of behavioral signals in improving recommendation

relevance, and the robustness of the linear bandit approach across different evaluation paradigms. The reward function design successfully captured user engagement patterns, with watchlist interactions providing the strongest positive signal and pagination indicating recommendation failures.

From a performance perspective, the Q-learning agent also showed promising early results when key hyperparameters like learning rate and decay rate were tuned appropriately. The increased learning rate, in particular, enabled the agent to reach higher rewards more quickly. However, the agent's precision ultimately plateaued, indicating a limitation in sustaining exploration and learning beyond the initial phases. This suggests that while Q-learning is capable of quick early learning, it may struggle to discover and exploit more complex and long-term strategies without additional support. For future iterations, incorporating techniques such as epsilon-greedy scheduling, exploration bonuses, or prioritized experience replay could help overcome the plateau and push the agent towards higher convergence and generalization.

G. Algorithm Comparison and Discussion

The evaluation revealed distinct performance characteristics across our three reinforcement learning approaches. The contextual bandit demonstrated superior immediate reward optimization while maintaining reasonable exploration behavior. Q-Learning provided interpretable decision-making through explicit state-action value functions but faced scalability challenges with the large state space. The Deep Q-Network approach showed promise for handling complex feature interactions through its neural architecture, though it required more extensive hyperparameter tuning and computational resources.

The contextual bandit's linear model architecture proved particularly effective for the course recommendation domain, successfully capturing the relationship between student features and course preferences without overfitting. The 260-dimensional feature space provided sufficient expressiveness while maintaining computational efficiency for real-time recommendation scenarios.

Compared to more advanced reinforcement learning algorithms, Q-learning demonstrated a relatively straightforward learning process, particularly when coupled with a carefully designed reward function. Nevertheless, the agent exhibited a parabolic trend in its precision curve, signaling challenges with convergence over longer episodes. This limitation, while expected, highlights Q-learning's inherent struggles with balancing exploration and exploitation, particularly in large or complex state spaces. To address this, future work could consider integrating more sophisticated exploration techniques such as softmax action selection, or even more sophisticated decaying epsilon strategies to help the agent navigate complex environments more effectively.

V. CONCLUSIONS

Our implementation and evaluation of reinforcement learning approaches for course recommendation demonstrates the viability of contextual bandits for personalized educational guidance systems. The contextual bandit approach successfully learned from user engagement signals while maintaining reasonable exploration behavior and course coverage. The comprehensive evaluation framework validated algorithm performance across multiple methodologies, providing confidence in the robustness of our findings.

The 260-dimensional feature representation proved effective for capturing student context, while the linear model architecture provided adequate expressiveness for the recommendation task. Key design decisions including the reward function structure, exploration strategy, and feature engineering contributed to successful learning outcomes. The A/B testing revealed important trade-offs between immediate performance and long-term discovery, informing parameter selection for different deployment scenarios.

Future work should explore non-linear model architectures to capture more complex feature interactions, investigate curriculum learning approaches for improved cold-start performance, and develop more sophisticated exploration strategies that account for course prerequisites and academic progression requirements. Integration with real user data and deployment in the OpenMario environment will provide opportunities for online learning and continued algorithm refinement.

REFERENCES

[1] T. Xiao and D. Wang, "A General Offline Reinforcement Learning Framework for Interactive Recommendation." Accessed: May 05, 2025. [Online]. Available: https://arxiv.org/abs/2310.00678

[2] J. Zhou and E. Agichtein, "RLIRank: Learning to Rank with Reinforcement Learning for Dynamic Search," *arXiv (Cornell University)*, pp. 2842–2848, 2020, doi: 10.1145/3366423.3380047.

[3] Z. Xu, A. Tran, T. Yang, and Q. Ai, "Reinforcement Learning to Rank with Coarse-grained Labels." Accessed: May 05, 2025. [Online]. Available: https://arxiv.org/abs/2208.07563