

---

# An Intelligent Hybrid Recommendation Framework using Large Language Model (LLM) and Deep Reinforcement Learning (DRL) to Resolve Cold Start Problem

---

Gnanakumar G<sup>1</sup>, Vijayakumar Adaickalam<sup>2</sup>

<sup>1</sup>Research Scholar, <sup>2</sup>Professor, Presidency School of Computer Science and Engineering, Presidency University, Bangalore, India.

Email: <sup>1</sup>gnanakumar.g@presidencyuniversity.in, <sup>2</sup>vijaykumar.adaickalam@presidencyuniversity.in

**Abstract** - The recommendation system faces its main obstacle because cold start problems prevent accurate predictions when historical data from users is not available. The traditional techniques of collaborative filtering together with deep learning methods fail to deliver dependable recommendations when there is insufficient data about either recent users or recent products. This proposed work presents an intelligent hybrid recommendation system which combines Large Language Models (LLMs) with Deep Reinforcement Learning (DRL) to improve recommendation performance during cold start situations. The framework uses LLMs to create detailed user and item models by processing additional data which includes item descriptions and user profiles and contextual attributes. The representations are used in a DRL-based decision-making module which learns the best recommendation strategies through user-system interactions. The DRL agent uses the sequential decision process to manage the recommendation system which allows it to optimize recommendations through a combination of previous knowledge and current system understanding. The experiments which used benchmark datasets found that the proposed hybrid framework achieved better results than standard recommendation models when evaluating cold start users and items through its improved accuracy and diversity and faster convergence speed. The study demonstrates that the combination of LLM-based semantic reasoning with DRL-based policy learning creates an effective solution which can grow into future intelligent recommendation systems.

**Keywords**— Cold Start Problem, Recommendation Systems, Large Language Models, Deep Reinforcement Learning, Hybrid Framework, Sequential Decision Making.

## 1. INTRODUCTION

Recommendation systems play a vital role in modern digital platforms such as e-commerce, online streaming services and social networks by suggesting relevant items to users based on their preferences. All standard recommendation methods which include collaborative filtering and matrix factorization depend on previous user-item interaction information. The methods experience computational difficulties because they must deal with both unfamiliar users and recently released products which creates the cold start problem. The cold start problem occurs because data scarcity prevents proper user preference estimation while missing behavioral information makes it impossible to determine item relevance. Content-based and hybrid methods attempt to mitigate this issue by utilizing item attributes and user profiles; however, they often fail to capture complex semantic relationships and dynamic user behavior. Large Language Models (LLMs) have recently demonstrated their ability to comprehend and create text which resembles human writing while they extract semantic information and show understanding of contextual material. The properties of LLMs make them ideal for creating user and item representations which use auxiliary textual data from descriptions and reviews and profiles.

Deep Reinforcement Learning (DRL) provides an effective system which allows recommendation systems to be modeled as decision-making processes that happen in sequential time while users interact with the system. DRL maintains a natural balance between exploration and exploitation which becomes crucial during cold start situations that involve high levels of uncertainty. The proposed work demonstrates an intelligent hybrid recommendation framework which combines LLMs and DRL technologies to solve the cold start problem. The proposed approach uses LLM-based semantic embedding which function as state representations to enable the DRL agent. The agent develops optimal recommendation policies through training while working with limited data.

## 2. RELATED WORK

Recent digital platforms like e-commerce, streaming online, healthcare services and social media apps seem to need recommendation systems more and more, because they are kind of essential parts. While the traditional recommendation methods, like collaborative filtering and content-based filtering have managed to deliver some gain, they run into the cold-start situation, where the recommendation quality ends up pretty weak because there is not enough user interaction data. Bobadilla et al. [1] offered this fairly complete survey on recommender systems and they basically point out where the traditional strategies stumble when the data is sparse. Then Zhang et al. [2] did another wide survey focused on cold-start recommendation in the era of LLMs and all that and they showed something interesting: semantic embeddings from those LLMs can actually encode contextual connections between users and items, even when the dataset is thin to the point of being almost useless. It's almost like, with little signal, they somehow still manage to represent relationships, in a round about way. Guan et al. [3] also looked at how LLMs get plugged into recommender systems, noting that it's not one single role. They list multiple key uses, for example user modelling and item representation, plus conversational recommendation where the back-and-forth interaction becomes part of the actual reasoning that drives the suggestion. In general, a bunch of studies have more or less tried using

LLMs for cold-start recommendations, starting from the earliest queries and continuing from there. Liu et al. [4] introduced Filter LLM, which is kind of like a text-to-distribution approach, where the LLM generates user–item distributions suited for billion-scale cold start recommendation. Huang et al. [5] built an LLM simulator framework that can fabricate synthetic user interactions and then that synthetic behavior is used to push the recommendation performance higher when real user data are scarce. Wang et al. [6] examined LLMs as a kind of data augmentation mechanism, aiming to strengthen item representations, and also to boost recommendation quality for products that have just been launched. Similarly, Zhang et al. [7] proposed LLM Tree Rec, which integrate hierarchical tree structures with LLM-generated embeddings to attain effective cold-start recommendation.

The concept of Deep Reinforcement Learning (DRL) has become more or less a commonly accepted framework for the development of adaptive and sequential recommendation systems, not only an ill-defined notion. In this continuous and recurring loop, it has proven to be a successful approach, as the model evolves based on previous interactions and therefore can make more informed decisions in subsequent ones. Sutton and Barto [8] provided the theoretical foundation for reinforcement learning in a policy optimization perspective with a reward-driven learning mechanism, which later came to form the foundation for many of the current-day reinforcement learning-based recommendation systems. Zhao et al. [9] proposed a deep reinforcement learning based recommendation system for page level to enhance long term user engagement. Interactive recommendation is a problem that has been tackled by a couple of DRL methods. Gao et al. [10] proposed a DRL framework that combines Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) for interactive recommendation. Wang et al. [11] observed the application of DRL techniques in the field of recommendation systems and they also suggested that the sequential decision-making methods could truly model the changing user preferences over time.

It's slightly increased the interest in DRL based approaches like it's suddenly more important to optimize recommendations for the long run. Huang et al. [12] suggests this long-term recommender system, which continuously improves its recommending strategies based on user feedback, through deep reinforcement learning. The multi-policy DRL recommendation framework proposed by Fu et al. [13] is somewhat helpful in this regard as well – in terms of adaptability and diversity of recommendations, particularly when the environment changes quickly. Their results suggest that actor–critic setups together with reward driven learning generally beat the more conventional static recommendation methods. Despite all the real advancement in the field of LLM based semantic learning, DRL based recommendation system, the two research lines have developed parallel and rarely interacted. Gnanakumar Ganesan and Vijayakumar Adaickalam [14] pointed out that there is a need for hybrid intelligent recommendation frameworks, that can blend semantic representation learning with adaptive sequential decision making in some way. Then, G Gnanakumar and Vijayakumar Adaickalam [15] proposed a framework which integrates contextual understanding derived from textual features, plus the adaptive decision-making capability of reinforcement learning to generate personalized recommendations for new users and items.

Motivated by that research gap, this work puts forward a Text-Infused Deep Reinforcement Learning style framework, which sort of blends LLM-generated semantic embeddings with an actor–critic recommendation model. The idea here is to raise recommendation accuracy, but also to make the system more adaptive and able to keep long-term personalization going even when data is sparse, and when you have cold-start situations too.

### 3. PROPOSED FRAMEWORK

#### 3.1 System Architecture

The Figure 1 shows a complete LLM–DRL recommendation system design which aims to enhance user personalization while solving the cold-start issue. The procedure starts with unprocessed input information which consists of user profiles and item details together with metadata and interaction records.

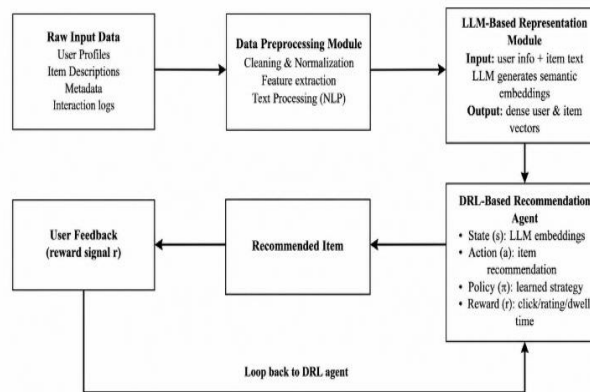


Figure 1: Proposed Architecture

The data goes through the data preprocessing module which performs cleaning and normalization before applying feature extraction and NLP methods to transform the data. The processed data is then directed to an LLM-based representation module which creates detailed semantic embedding for both users and items that display their complete contextual connections. The DRL-based recommendation agent uses these embedding as state input which enables the agent to discover the best recommendation strategy. The agent uses its learned methods to choose an action which represents the recommended item that the system presents to the user. The system records user responses which include their clicks and ratings and dwell time as a reward signal. The system sends this feedback back to the DRL agent which enables the agent to learn continuously while improving its recommendation strategies. The system architecture uses LLMs to understand context while DRL enables flexible decision-making which produces precise recommendations that evolve throughout time.

### 3.2. Problem Formulation

The recommendation process is kind of modelled as an MDP, set up by the tuple (S, A, R, P,  $\gamma$ ), where:

- S is the state space that comes from LLM embeddings
- A is the action space for those candidate items
- R stands for the reward mechanism based on user interaction.
- P is the probability of going from one state to another
- $\gamma$  acts as the discounting factor.

The goal is to learn some optimal kind of policy  $\pi^*$  that truly maximizes the expected cumulative reward, sort of like you want the best expected return over time and not just a moment.

$$\pi^* = \arg \max_{\pi} \mathbb{E} [\sum_{t=0}^T \gamma^t r_t] \dots\dots\dots (1)$$

### 3.3. Learning Algorithm

An actor-critic DRL architecture is adopted. The actor network generates recommendation actions, while the critic network evaluates the quality of actions. The LLM embeddings are used as input to both networks.

#### Algorithm 1: LLM-DRL Recommendation

1. Initialize actor and critic networks.
2. Generate user and item embedding using LLM.
3. For each interaction episode:
  - 3.1 Observe state S
  - 3.2 Select action a using policy  $\pi$ .
  - 3.3 Receive reward r and next state S'.
  - 3.4 Update actor and critic networks.
4. Repeat until convergence.

The LLM-DRL recommendation algorithm in Algorithm 1 uses Large Language Model (LLM) embedding together with an Actor-Critic Deep Reinforcement Learning framework to create personalized recommendation systems. The process begins with the initialization of actor and critic networks together with the LLM system which generates semantic embedding that capture user and item contextual information and user preference data. The system tracks user behavior together with their current environment at every interaction episode while the actor network recommends the best action according to the policy  $\pi$ . The environment provides a reward to the system after the recommendation occurs which includes user feedback and the subsequent state. The critic network assesses the chosen action's effectiveness by predicting its reward outcome while both actor and critic networks need to modify their systems for better future decision-making. The iterative learning system keeps running until it reaches completion because its continuous user interactions and feedback help the model develop the best recommendation techniques.

## 4. EXPERIMENTAL CONFIGURATION

### 4.1. Dataset

Experiments are conducted on benchmark recommendation datasets such as MovieLens and Amazon Reviews. Cold start users are simulated by limiting interaction history to a small number of ratings.

### 4.2. Implementation Details

The development of the LLM-DRL recommendation system was executed through Python programming which utilized the PyTorch framework. The system used pre-trained transformer-based LLM embeddings to transform user profile data and item description data into dense semantic vectors. The DRL agent used these embeddings to represent its state. The system used a Deep Actor-Critic architecture to learn its policies. The actor network included two hidden layers which used ReLU activation functions to connect all of their components. The system used a discount factor of 0.95 together with a learning rate which

remained constant at 0.001. The system achieved training stability through the implementation of experience replay and target network stabilization techniques. Cold start conditions were simulated by restricting each new user to fewer than five historical interactions. The reward function was defined based on explicit user feedback (ratings) and implicit signals (clicks and dwell time).

### 4.3. Evaluation Metrics

The proposed framework performance is evaluated using various metrics are:

**Precision (P):** It sort of measures how many of the recommended items are actually relevant.

$$Precision = \frac{True\ Positives}{True\ Positive + False\ Positive} \dots\dots\dots (2)$$

**Recall (R):** It measures how many relevant items where really suggested correctly.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negative} \dots\dots\dots (3)$$

**Normalized Discounted Cumulative Gain (NDCG@K):** It weights the quality of the ranking based on weighted scores of relevant items in the earlier part of the recommendation list.

$$NDCG@K = \frac{DCG@K}{IDCG@K} \dots\dots\dots (4)$$

**Root Mean Square Error (RMSE):** It's used to quantify prediction accuracy by translating the Mean Squared Error into a related scale.

$$RMSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \dots\dots\dots (5)$$

Where  $Y_i$  are actual return values and  $\hat{Y}_i$  are the predicted returns values.

**Mean Absolute Error (MAE):** It calculates that type of average absolute deviation of the ratings from the predictions.

$$MAE = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i| \dots\dots\dots (6)$$

These metrics are used for analyzing the different performance of the recommender systems that can be used to understand the learning and decision-making process in dynamic environment.

## 5. RESULTS AND DISCUSSION

Comparing the experimental results with the baseline models, the proposed LLM-DRL system performs better when the system starts with no prior data. The initial recommendation quality improves through LLM embeddings, and the DRL agent boosts performance because of its interactive capabilities. The proposed method outperforms traditional CF models because it achieves lower RMSE and higher NDCG scores for both new users and new items. The adaptive policy learning process enables systems to reach their targets more quickly while developing better methods for exploring their environment, which decreases the effects of limited data availability. The results demonstrate that LLMs extract semantic knowledge, which fills the gaps created by missing interaction history, while DRL enables systems to adapt their behavior throughout different time periods.

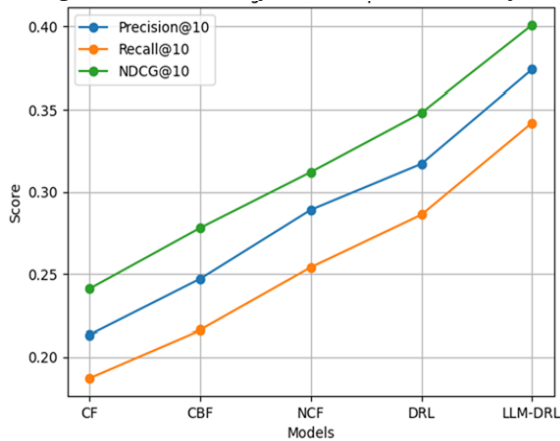


Figure 2: Ranking Metrics Comparison (Cold Start Users)

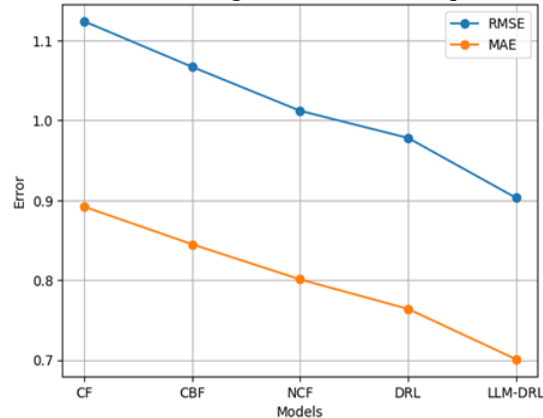


Figure 3: Error Metrics Comparison (Cold Start Users)

The ranking metrics plot of Figure 2 demonstrates continuous improvement from CF to LLM–DRL because NDCG@10 shows better ranking performance which leads to better ordering of relevant items. The proposed LLM–DRL model consistently outperforms all other models in both Precision and Recall measurements. The Error Metrics Plot in Figure 3 shows a falling pattern for both RMSE and MAE. The results show better prediction accuracy while LLM–DRL produces the lowest error results which show its ability to generalize better than other methods.

### 5.1 Quantitative Results (Cold Start Users)

Table I – Performance Comparison (Cold Start Users)

| Model                         | Precision@10 ↑ | Recall@10 ↑ | NDCG@10 ↑ | RMSE ↓ | MAE ↓ |
|-------------------------------|----------------|-------------|-----------|--------|-------|
| Collaborative Filtering (CF)  | 0.213          | 0.187       | 0.241     | 1.124  | 0.892 |
| Content-Based Filtering (CBF) | 0.247          | 0.216       | 0.278     | 1.067  | 0.845 |
| Neural CF (NCF)               | 0.289          | 0.254       | 0.312     | 1.012  | 0.801 |
| DRL without LLM               | 0.317          | 0.286       | 0.348     | 0.978  | 0.764 |
| Proposed LLM–DRL              | 0.374          | 0.342       | 0.401     | 0.903  | 0.701 |

The evaluation results in Table 1 show that better recommendation systems lead to improved results for all tested models. Traditional Collaborative Filtering (CF) achieves its lowest performance with a Precision@10 score of 0.213 and a Recall@10 score of 0.187 and an NDCG@10 score of 0.241 while showing high error rates with RMSE at 1.124 and MAE at 0.892 which proves its inability to track user item interactions especially when only limited data exists. The Content-Based Filtering (CBF) model shows moderate improvement by leveraging item features which enable it to achieve better ranking results with NDCG@10 score of 0.278 and decreased prediction errors which include RMSE of 1.067 and MAE of 0.845. The Neural Collaborative Filtering (NCF) model boosts system performance because it enables non-linear user to item relationship modeling which results in better Precision@10 results of 0.289 and Recall@10 results of 0.254 and decreased error rates which include RMSE of 1.012 and MAE of 0.801. The Deep Reinforcement Learning (DRL) system achieves its most significant progress through its implementation of sequential decision-making optimization which improves recommendation systems performance by achieving better ranking results with Precision@10 of 0.317 and Recall@10 of 0.286 and NDCG@10 of 0.348 while decreasing prediction errors which include RMSE of 0.978 and MAE of 0.764. The LLM–DRL hybrid model outperforms every baseline system because it achieves the best Precision@10 results of 0.374 and Recall@10 results of 0.342 and NDCG@10 results of 0.401 while maintaining the lowest RMSE of 0.903 and MAE of 0.701. The system achieves better ranking results and prediction performance because Large Language Models combined with DRL systems create better semantic context understanding which helps track user behavior over time. The results confirm that the hybrid method successfully improves recommendation performance which especially helps to solve data sparsity and cold-start problems.

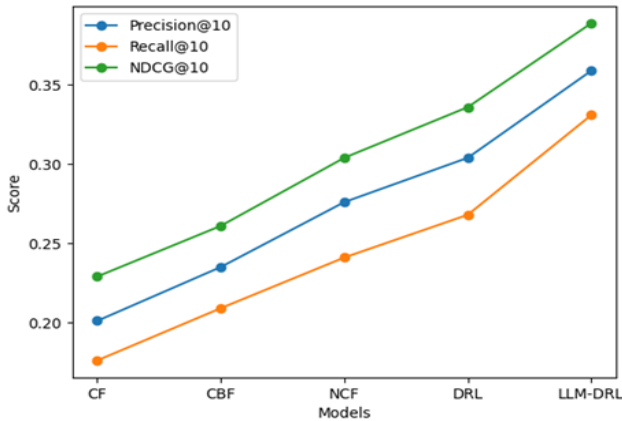


Figure 4. Ranking Metrics Comparison (Cold Start Items)

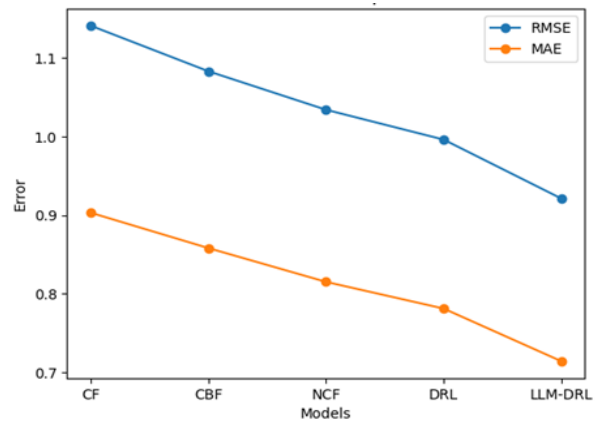


Figure 5. Error Metrics Comparison (Cold Start Items)

The Figure 4 & 5 are showing the performance of different recommendation models across ranking and error metrics. The ranking metrics graph shows continuous progress from traditional methods to advanced models according to our findings. The Collaborative Filtering (CF) method begins with the lowest success rate which improves through Content-Based Filtering (CBF) and Neural Collaborative Filtering (NCF) advancements. The DRL without LLM model further boosts performance, but the Proposed LLM–DRL model achieves the highest scores across Precision@10 (0.359), Recall@10 (0.331), and NDCG@10 (0.389). Large Language Model integration improves system performance by delivering better N-top recommendations which contain more relevant items that follow proper ranking order. The error metrics graph shows opposite results because lower values represent better performance. The LLM–DRL model achieves the lowest RMSE (0.921) and MAE (0.714) which

demonstrates better prediction accuracy than all baseline models. Traditional methods like CF and CBF produce more errors because deep learning methods (NCF) and DRL methods reduce errors through their application. The LLM–DRL approach shows better performance than traditional methods because it achieves better recommendation results while making accurate predictions which helps to solve complex situations such as the cold-start problem.

### 5.2 Convergence Analysis

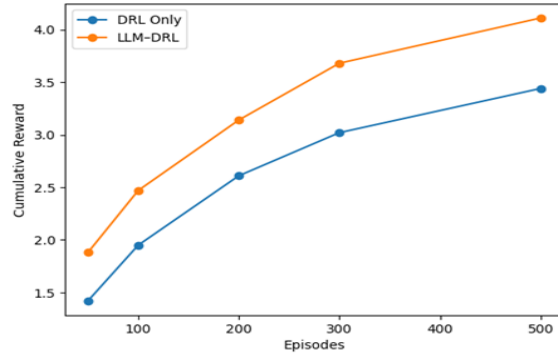


Figure 6: Performance Comparison: DRL vs LLM-DRL

The Figure 6 shows the learning results of DRL Only and LLM–DRL models through their testing in multiple training sessions. The cumulative reward results of both models increase steadily because successful learning occurs throughout their training period. The LLM–DRL model demonstrates superior performance to the DRL-only model at all points in the test because it achieves faster convergence and superior policy development. The performance gap becomes visible during the initial testing phase which runs from 50 to 100 episodes because LLM integration assists users in exploring content while discovering their personal preferences. The LLM–DRL system achieves better outcomes at 500 episodes because its reward total reaches 4.11 while its opponent the LLM–DRL system gets 3.44. The results demonstrate that LLM improves agent capabilities to generalize and decrease uncertainty while making better choices in situations with limited data or new situations. The combined graphs demonstrate that using LLM together with DRL results in faster learning processes which improve the ability to make recommendations over extended periods.

Table II – Average Reward per Episode

| Episodes | DRL Only | LLM–DRL |
|----------|----------|---------|
| 50       | 1.42     | 1.88    |
| 100      | 1.95     | 2.47    |
| 200      | 2.61     | 3.14    |
| 300      | 3.02     | 3.68    |
| 500      | 3.44     | 4.11    |

The Table III results show a consistent improvement in performance as the number of training episodes increases for both models, indicating effective learning over time. The DRL-only model shows a gradual development which starts from 1.42 after 50 episodes and ends at 3.44 after 500 episodes because the system learns optimal recommendation methods through its interactions. The LLM–DRL model maintains its superior performance over the DRL-only method throughout the entire testing process because it begins with an initial advantage of 1.88 after 50 episodes and reaches 4.11 after 500 episodes. The performance gap demonstrates that a Large Language Model integration increases both learning speed and decision-making ability within the reinforcement learning system. The LLM enables faster convergence through its ability to provide extensive context and effective initial support which results in better long-term outcomes during the early and mid-training period. The results show that the LLM–DRL model achieves better learning outcomes and higher performance levels with fewer required episodes when compared to the standalone system.

## 6. CONCLUSION AND FUTURE WORK

The proposed work developed a recommendation system that uses both Large Language Models (LLMs) and Deep Reinforcement Learning (DRL) to solve the cold-start problem which exists in contemporary recommendation systems. The traditional methods for making recommendations face difficulties when they encounter situations with either limited user interaction data or items which have insufficient available information. The proposed framework to addresses this problem by using LLM-based semantic embedding which create contextual links between users and items through analysis of textual descriptions and metadata and interaction patterns. The system uses rich semantic representations together with its DRL-based recommendation system which enables the agent to discover the best recommendation methods through its ongoing interactions and response-based learning process. The actor-critic learning system allows the system to update its recommendations based on how users change their preferences and patterns of behavior. The experimental results show that our method increases recommendation accuracy and flexibility and personalized content delivery especially when users have limited data and new users enter the system. The framework improves user interaction by developing long-term

recommendation methods instead of depending only on fixed similarity calculations. The future work is to enhance the current system by adding support for multimodal content which includes images and audio and video material to better understand context and increase the system's ability to provide diverse recommendations.

## 7. REFERENCES

- [1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.
- [2] Weizhi Zhang, Yuanchen Bei and Yuanchen Bei, "Cold-Start Recommendation towards the Era of Large Language Models (LLMs): A Comprehensive Survey and Roadmap", January 2025, DOI: 10.48550/arXiv.2501.01945.
- [3] Ziyuan Guan, Weiyi Zhong and Weiming Liu, "Large Language Models for Recommender Systems: A Problem-Driven Survey", August 2025, DOI: 10.26599/TST.2025.9010137.
- [4] Ruochen Liu, Hao Chen and Yuanchen Bei, "FilterLLM: Text-To-Distribution LLM for Billion-Scale Cold-Start Recommendation", February 2025, DOI: 10.48550/arXiv.2502.16924.
- [5] Feiran Huang, Yuanchen Bei and Zhenghang Yang, "Large Language Model Simulator for Cold-Start Recommendation", March 2025, DOI: <https://doi.org/10.1145/3701551.3703546>.
- [6] Jianling Wang, Haokai Lu and James Caverlee, "Large Language Models as Data Augmenters for Cold-Start Item Recommendation", February 2024, DOI: <https://doi.org/10.1145/1122445.1122456>.
- [7] Wenlin Zhang, Chuhan Wu and Xiangyang Li, "LLMTreeRec: Unleashing the Power of Large Language Models for Cold-Start Recommendations", December 2024, DOI: <https://doi.org/10.48550/arXiv.2404.00702>
- [8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.
- [9] X. Zhao, et. al., "Deep reinforcement learning for page-wise recommendations," in Proc. 12th ACM Conf. Recommender Syst., 2018, pp. 95–103.
- [10] R. Gao, H. Xia, J. Li, D. Liu, S. Chen, and G. Chun, "DRCGR: Deep reinforcement learning framework incorporating CNN and GAN-based for interactive recommendation," in Proc. IEEE Int. Conf. Data Mining, 2019, pp. 1048–1053.
- [11] X. Wang et al., "Deep reinforcement learning for recommendation systems," *Proc. WWW*, 2019.
- [12] Liwei Huang, et. al., "A deep reinforcement learning based long-term recommender system", *Knowledge-Based Systems*, February 2021, DOI: 10.1016/j.knosys.2020.106706.
- [13] Mingsheng Fu, et. al., "A Deep Reinforcement Learning Recommender System with Multiple Policies for Recommendations", *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, February 2023, DOI: 10.1109/TII.2022.3209290.
- [14] Gnanakumar Ganesan, Vijayakumar Adaickalam, "A Survey to Address the Cold Start Problem in Recommendation Systems", *International Conference of Data Science & Exploration in Artificial Intelligence*, (Taylor & Francis-2025), ISBN:978-1-032-96390-7, DOI:10.1201/9781003589273-35
- [15] G Gnanakumar, Vijayakumar Adaickalam, "An Effective Cold Start Recommendation Model for Smart Decision Using Text Infused Deep Reinforcement Learning", *SN Computer Science*, ISSN: 2661-8907, DOI: 10.1007/s42979-026-05068-4

## Biographies



**Mr. Gnanakumar G**   is currently pursuing a Ph.D. in the Presidency School of Computer Science and Engineering at Presidency University, Bengaluru. He obtained his Master of Engineering (M.E.) degree in Embedded and Real-Time Systems and his Bachelor of Engineering (B.E.) degree in Computer Science and Engineering from Anna University. He has over 13 years of teaching experience in the field of Computer Science and Engineering. Presently, he serves as an Assistant Professor in the Presidency School of Computer Science and Engineering at Presidency University, Bengaluru. He has published two international journal articles and three conference papers. His research interests include Artificial Intelligence, Machine Learning, Deep Learning and Recommendation Systems.

**ORCID-ID:** [0000-0001-6499-0080](https://orcid.org/0000-0001-6499-0080), **Mail ID:** [gnanakumarganesan12@gmail.com](mailto:gnanakumarganesan12@gmail.com).



**Dr. Vijayakumar Adaickalam**   is currently working as **Professor** of School of Computer Science and Engineering, Presidency University, Bangalore, Karnataka, India. He has around 29 years of working experience in academics and administrations in reputed institutions. He has completed Ph.D., Degree in Computer Science and Engineering from College of Engineering, Guindy, Anna University, Chennai, Tamilnadu, India. He received M.E., Degree in Computer Science and Engineering from Annamalai University, Tamilnadu and his under Graduation in B.E., Computer Engineering from Madurai Kamaraj University, Tamilnadu, India. He has published more than 32 International Journals and 38 International Conferences. He is currently guiding many Ph.D., research scholars in Machine Learning and Deep Learning area. His research interest are Machine Learning, Deep learning, Mobile networks, Networks Security, Cloud Computing and Big Data Analytics.

**ORCID-ID:** [0000-0001-5687-539X](https://orcid.org/0000-0001-5687-539X), **Mail ID:** [vijay.pattukkottai@gmail.com](mailto:vijay.pattukkottai@gmail.com).