OPEN ACCESS
International Journal of Innovations in Science & Technology
RESEARCH & INNOVATION DIVISION
IJIST

Leveraging Clustering and Reinforcement Learning for a Product Recommender System

Mubbashir Ayub[1]*, Amna Obaid[2], Tasawer Khan[2]
[1]Department of Software Engineering, University of Engineering and Technology, Taxila, Pakistan
[2]School of Information Technology, Whitcliffe Auckland Campus, New Zealand
***Correspondence**: mubbashir.ayub@uettaxila.edu.pk

Citation | Ayub. M, Obaid. A, Khan. T, "Leveraging Clustering and Reinforcement Learning for a Product Recommender System", IJIST, Vol. 8 Issue. 4 pp 1660-1681, July 2026
Received | June 15, 2026 **Revised |** July 12, 2026 **Accepted |** July 15, 2026 **Published |** July 19, 2026.

The rapid development of digital platforms and e-commerce necessitates the adoption of strong recommender systems that can accommodate a wide range of user preferences. Traditional recommender systems often rely on collaborative filtering and content-based methods, which cannot grasp the complex relationships between users and products. Systems struggle with issues including sparse data, little flexibility, and inadequate user customization. The static nature of collaborative filtering and content-based filtering restricts their efficacy in dynamic situations, where item availability and user preferences can change. Here we are proposing a reinforcement-learning-based method that operates on the clustered data, where the agent learns optimal strategies to recommend products by navigating a defined action space. The proposed method employs a grid-based environment where actions determine the agent's progression through clusters. A variety of clustering algorithms were applied, and the one with the minimum standard deviation was selected for the formulation of states of a grid-based environment. Rewards are assigned based on user satisfaction with the recommended items, encouraging the agent to make relevant and impactful recommendations. Evaluation is done on two public datasets, Amazon and Movie Lens. The evaluation parameters used were start state count analysis, number of states visited by each user, return earned to reach a goal state from a predetermined start state, and number of steps taken in each episode during the learning process.

Keywords: Clustering, Policy, Q-Learning, Markov Decision Process, Standard Deviation and K-Means

Introduction:

A product recommender system is a specific type of recommender system that focuses on suggesting products to users based on their preferences, behavior, and other relevant data. These systems are commonly used in e-commerce platforms to enhance user experience by offering personalized product recommendations [1]. Conventional recommendation techniques involve Collaborative Filtering (CF) and Content Based Filtering (CBF). Clustering is used to group related items together and dissimilar ones into different clusters. In recommendation systems, comparable users can be grouped together based on their interests and behaviors by using clustering algorithms [2]. This information can then be used to provide customized suggestions depending on the predilections of the cluster to which all users belong. Reinforcement learning (RL) is a machine learning paradigm where objective is to achieve long term user satisfaction through highly personalized recommendations. RL-based recommendation systems are being used in various domains including news article recommendation, job recommendations, product recommendations, movie recommendations and so on [3]. The combination of clustering and reinforcement learning (RL) in the development of product and movie recommender systems, is an emerging area of study, with significant implications for enhancing personalization and consumer satisfaction [4].

Our goal in this work is to contribute to the knowledge in the domain of recommender systems by demonstrating the practical benefits and limitations of combining clustering algorithms with RL in real-world applications, particularly in the E-commerce domain. Clustering is used to group similar users and items, simplifying the recommendation problem by reducing the complexity of the interaction space [5]. This step also enhances scalability by managing the large-scale data typically encountered in e-commerce platforms. An RL framework is designed to operate on the clustered data, where the agent learns optimal strategies to recommend items by navigating a defined action space [6]. The system employs a grid-based environment where actions (e.g., UP, DOWN, LEFT, RIGHT) determine the agent's progression through clusters. Rewards are assigned based on user satisfaction with the recommended items, encouraging the agent to make relevant and impactful recommendations. The system's performance is evaluated against two standard datasets for user engagement, conversion rates, and customer satisfaction.

Novelty of the proposed work:

Our work differs from prior clustering-assisted RL recommenders (e.g., Bi-clustering + MDP approaches such as [4] and [6]) in three ways. First, cluster membership itself — not a hand-designed similarity table — defines both the Q-learning state space and the reward signal, via a single Jaccard-similarity primitive used consistently for start-state selection (Eq. 1) and step reward (Eq. 1). Second, the clustering stage is evaluated against eight clustering algorithms using standard deviation as validity criteria, rather than assuming a single algorithm a priori. Third, the resulting method is evaluated across two structurally different domains (sparse e-commerce interactions vs. dense entertainment ratings), which lets us report where the approach is data-limited (Amazon) versus where it is data-sufficient (MovieLens) — a distinction most single-dataset clustering + RL recommender papers do not surface.

Literature Review:

CF is a recommendation system technique that predicts an individual's preferences based on similar user preferences. It uses data from interactions with a particular item, such as ratings or purchasing history, to identify similar users and make recommendations. There are two main approaches: user-based CF, which recommends items preferred by similar users, and item-based CF, which finds similar items to the target user's liking and recommends them [7]. CBF is a method that predicts a user's preferences based on the content of past preferences, such as genre, director, or actor. It applies to various items with attributes like books, movies, music, and products. The algorithm uses user's past behaviour to build a profile

of preferences, recommending items with similar attributes in the future [8]. Hybrid systems combine CF and CBF to leverage the strengths of both methods while mitigating their weaknesses [9]. Although hybrid systems often perform better and their accuracy is claimed to be higher than that of other single-method systems, they can be complex and computationally expensive.

Clustering Techniques in Recommender System:

Recommendation algorithms are a need in everything from video streaming services to e-commerce websites. They are employed to forecast a user's preferences based on their past interactions and comparable users' behaviors. Using clustering algorithms in recommendation systems is one approach to accomplish this. K-means clustering is a widely used unsupervised machine learning technique for dividing datasets into discrete groups or clusters. Machine learning (ML) is a sharply evolving area of artificial intelligence (AI) that concentrates on enabling computers for learning relationships and patterns from data as well as making decisions or predictions with human intervention minimally. ML models can expand their performance over time by large-scale data, optimization techniques, and leveraging statistical methods.

Advanced product recommender systems sharply depend on complex machine learning (ML) methods for capturing convoluted user behaviors, large-scale data patterns, and dynamic preferences. Previous studies have proved the usefulness of machine learning and deep learning (DL) models crosswise various domains, such as anomaly detection in real-time network traffic [10], predictive modelling in healthcare applications [11][12], and use of hybrid neural networks for sales forecasting in retail environments [13]. These research focused on ML models capability for learning from high-dimensional datasets, deliver accurate, heterogeneous, and real-time predictions. Likewise, sentiment analysis methods applied to social media, chatbot service encounters, and fake news detection underscore the status of extracting latent patterns from unstructured user-generated data, which is a critical component of personalized recommendation systems [14]. Moreover, system robustness, data privacy, and more importantly ethical considerations remain critical circumstances when implementing intelligent systems in real-world applications, especially those relating behavioral data and profiling of user [15].

Recent studies on ethical challenges in terms of data privacy and security vulnerabilities in platforms of social networking add significant insights relevant to recommender systems collecting as well as analyzing information of sensitive user [15]. Additionally, studies on the systems of real-time recognition including sign language alphabet recognition by means of various ML methods and optimizing different models for New Zealand Sign Language proves the status of continuous model improvement of real-time decision-making and adaptive learning [16][17]. These assumptions adjust with recommender systems of reinforcement learning-based, where people can learn strategies of optimal recommendation via interaction as well as feedback. By clustering methods integration for user segmentation including reinforcement learning for flexible recommendation policies, this research builds upon the foundations of ML, Establishment for proposing an intelligent, ethical as well as scalable product recommender framework.

The algorithm identifies patterns in data by grouping items into K-clusters while minimizing the sum of squared errors within each cluster. This method has been extensively applied in various domains, such as anomaly detection, image compression, and consumer segmentation, providing businesses with actionable insights. However, while K-means is effective in identifying patterns, its reliance on predefined K-values and sensitivity to initial centroids often limits its adaptability to dynamic and large-scale recommendation systems [18]. These limitations highlight the need for advanced approaches, such as reinforcement learning, which can dynamically model evolving user preferences and optimize recommendations

beyond static clustering techniques. The key advantage of K-means clustering is it's scalability as it performs well on large datasets, making it easier to apply for real-world applications. The interpretability of clustering method is easy to understand and interpret. It is computationally efficient algorithm specially for low-dimensional data.

A widely used method for arranging data into a hierarchy of clusters is hierarchical clustering, which can be achieved by combining smaller clusters into bigger ones (agglomerative approach) or by splitting larger clusters into smaller ones (divisive approach). Hierarchical clustering provides a structured and interpretable way to categorize users or objects in the context of recommendation systems, which can then allow more complex methods like reinforcement learning [19]. It improves the effectiveness of data analysis by simplifying data and grouping information according to criteria like past purchases, browsing habits, or product characteristics. Even while hierarchical clustering works well for organizing data, more research is still needed to fully understand how to combine it with reinforcement learning to produce highly personalized and dynamic suggestions. Identifying how these clusters can optimally interact with reinforcement learning agents to address challenges like scalability and real-time adaptability is a key gap in the current research [20].

A density-based clustering algorithm called DBSCAN was created to find outliers, or low-density points or noise, in datasets as well as clusters of different forms. In situations where the number of clusters is unclear, it is very beneficial. DBSCAN offers a versatile clustering framework by locating core points, joining them to create clusters, and separating noise points [21]. DBSCAN is especially well-suited for real-world datasets, such those found in recommendation systems, because it can detect irregularly shaped and unevenly dispersed clusters, unlike K-means clustering, which presupposes a predefined form for clusters. Because of its versatility, it can identify intricate patterns in user and product data that conventional clustering techniques can miss. Despite its benefits, DBSCAN's inability to handle datasets with different densities suggests a possible area for further research in recommendation contexts. Combining DBSCAN with innovative techniques like reinforcement learning presents a chance to create personalized and adaptable recommender systems. This collaboration tackles the difficulty of identifying complex user-product links in sizable datasets, emphasizing the possibility of further study to improve clustering-driven recommendation systems.

An unsupervised machine learning method called mean-shift clustering is frequently used in data analysis to find clusters within datasets. By repeatedly moving data points in the direction of regions with the highest density, it uses a density-based algorithm to find high-density zones. It is a useful tool for exploratory data analysis since, in contrast to some clustering techniques, it does not require prior knowledge of the number of clusters. Mean-shift clustering offers a versatile method for identifying organic groupings in user or product data in the context of recommendation systems. This flexibility allows for dynamic user and product segmentation, which is very beneficial for improving recommender systems. Mean-shift clustering makes it easier to create customized and adaptable recommendation systems when paired with more sophisticated methods like reinforcement learning [22]. However, existing literature highlights shortcomings in its capacity to scale for enormous datasets and integrate with sequential decision-making frameworks—two essential components of contemporary, large-scale recommender systems. By addressing these issues, clustering-based techniques in recommendation systems may perform much better and be more applicable.

A popular probabilistic method for modelling complex data distributions, density estimation, and clustering is Gaussian Mixture Models (GMM). A weighted sum of several Gaussian distributions, each with its own mean, variance, and covariance matrix, is how a GMM represents a dataset. When compared to single Gaussian distributions, GMM's versatility enables it to identify more complex patterns in data. GMM has been used to simulate

overlapping user preferences in recommender systems, allowing for more precise user segmentation and clustering. Nevertheless, despite its advantages, GMM has trouble scaling and managing the kind of big, sparse datasets that are commonly seen in e-commerce platforms [23]. Furthermore, GMM improves the flexibility and customization of suggestions when combined with reinforcement learning (RL). However, there are still unanswered questions about how to best integrate GMM with RL, especially when it comes to dynamically adapting to shifting user behaviour in real time. As such, this is an area that needs more investigation in the design of advanced recommender systems.

Spectral clustering is a robust data segmentation method that creates clusters from related data points by utilizing a dataset's affinity matrix. It gets beyond the drawbacks of conventional clustering techniques like K-means by utilizing eigenvectors to separate data into discrete groups. Datasets having clusters of different forms and non-linearly separable data are especially well-suited for this method. A common technique for dividing up users or items is spectral clustering, which makes it possible to better comprehend the intricate correlations present in the data. The flexibility and customization of recommender systems are improved when paired with more sophisticated techniques like reinforcement learning. By continuously analysing user activity and adjusting to shifting preferences over time, this integration enables more accurate and dynamic suggestions [24]. According to recent research, spectral clustering has a lot of promise, but there are still obstacles in the way of fully utilizing it in large-scale, real-time recommendation systems, particularly when it comes to resolving problems like scalability and data sparsity.

The OPTICS (Ordering Points to Identify the Clustering Structure) algorithm, a density-based clustering technique intended to find clusters of different densities in huge datasets, is one of the clustering algorithms reviewed in this section. OPTICS offers more flexibility than conventional clustering algorithms like DBSCAN (Density-Based Spatial Clustering of Applications with Noise), which need a fixed density threshold to define clusters. Instead, it generates an ordering of the data points that reflects the dataset's natural clustering structure [25]. The literature demonstrates the use of OPTICS in a few fields, such as the clustering of user and product data. Furthermore, OPTICS's potential for personalized and adaptable suggestions is demonstrated by its integration with cutting-edge techniques like reinforcement learning in recommender systems. Research gaps are also identified in the paper, specifically regarding how issues like data sparsity and user engagement in real-world settings might be addressed by merging clustering approaches with RL.

BIRCH (Balanced Iterative Reducing and Clustering utilizing Hierarchies) is one of several clustering approaches utilized in recommendation systems, according to a survey of the literature. A compact representation of a cluster arranged in a balanced tree structure is provided by the Clustering Feature, which was introduced by BIRCH. Because of its ability to efficiently insert and merge new data points, this hierarchical structure is especially well-suited for big or streaming datasets. BIRCH uses a threshold distance to determine whether cluster is appropriate and clusters data sequentially. In big data contexts, it is a popular clustering technique because of its speed and scalability. BIRCH's ability to manage enormous volumes of user and product data allows recommender systems to provide tailored suggestions as data changes [26]. Furthermore, BIRCH improves the system's ability to adjust to shifting user behaviours by combining with reinforcement learning to provide precise and dynamic recommendations that raise user engagement and satisfaction. Significant shortcomings in conventional recommendation methods, like scalability and adaptability to real-time user interactions, are addressed by combination of clustering and reinforcement learning.

Reinforcement Learning in Recommender Systems:

Reinforcement learning (RL) teaches an agent to make decisions by interacting with its surroundings to maximize a cumulative reward. RL models learn from their own actions

and the input they receive, as opposed to supervised learning, which uses a fixed dataset. It allows recommender systems to learn and evolve in real-time, making them more responsive to changing user preferences and behaviours [27].

The key concept of RL includes:

Agent: The entity that makes decisions (e.g., the recommender system). It learns the optimal strategies to recommend products.

Environment: The space in which the agent operates (e.g., the platform where recommendations are made). The environment provides feedback to the RL agent in the form of rewards, which are used to adjust and improve the recommendation strategy.

Action: The steps taken by the agent in response to user behavior.

Reward: The feedback received after an action, indicating success or failure (e.g., a user clicking on a recommended product).

Policy: The RL agent updates its policy based on the reward feedback. The goal is to learn a policy that maximizes long-term rewards, meaning the system aims to make recommendations that lead to higher user engagement or satisfaction over time.

Exploration vs. Exploitation: In real-world (RL) systems, agents exploit knowledge to make optimal recommendations based on past user interactions. Exploration, on the other hand, involves exploring new patterns and user preferences. Balancing exploration and exploitation is crucial to avoid suboptimal recommendations and irrelevant suggestions.

Markov Decision Process (MDP) is an extension of Markov chains. MDP provides a framework for modelling decision making and is very useful for optimization problems. MDPs lay the ground for most RL algorithms as well. An MDP is described as a 4-tuple (S, A, Pa, Ra) [14].

$\boldsymbol{S}$ is a set of state space or state transition St.

$\boldsymbol{A}$ is a set of actions called the action space (alternatively, As is the set of actions available from state ss),

$\boldsymbol{Pa}$ $(s, s') = Pr\,(st{+}1 = s' \mid st = s, at = a)$ is the probability that action a in state time t will lead to a state s' at time $t + 1$,

$\boldsymbol{Ra}$ (s, s') is the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action a.

Markov Decision Processes (MDPs) are crucial to product recommender systems as they offer a formal framework for decision-making problems, they consist of incentives giving feedback, actions suggesting products, and states expressing user contexts [28]. By breaking down the state space, clustering facilitates RL and improves recommendation strategies. Adaptability and customisation are improved by combining MDP with clustering and reinforcement learning. Choosing reinforcement learning algorithms and specifying components, state space, actions, rewards, and transition dynamics are some of the implementation processes. Better personalization, scalability, and dynamic adaptation are among the advantages. MDPs are used to formalize the process of recommending products to users in a way that optimizes long-term user engagement, satisfaction, or other business objectives [29].

Clustering based RL approaches overcome the challenges faced by large datasets and changes in user behaviour. The key components involved in the proposed method include:

Cluster Initialization: Initially, characteristics such as product categories, purchase frequency, or user demographics are used to group users or products. This reduces the state space and simplifies the problem of recommendation.

Cluster-Level Policy Learning: By having RL agents learn policies for every cluster, the system may adjust suggestions based on the preferences of the group rather than generalized data points.

Dynamic Updates: Reinforcement learning policies are adjusted to consider updated information as well as frequent clustering updates in response to changing user preferences. This ensures that the system remains responsive to shifts in user behaviour over time.

This approach offers several advantages:

Enhanced Personalization: By using clustering, the system can provide more precise and pertinent recommendations by grouping individuals based on shared tastes.

Scalability: The RL model can scale more successfully, even in situations with a high volume of users and products, by utilizing clustering to reduce the complexity of the state and action spaces.

Enhanced Learning Efficiency: If the RL model is concentrated on a more homogeneous group of consumers or products, it may learn within clusters more quickly and efficiently. Faster convergence and improved overall performance may result from this.

Adaptability: The system may continuously evolve and adapt over time since dynamic clustering is included to ensure that it remains responsive to changes in user behaviour.

Despite the potential of clustering-based RL systems, there are several gaps in the literature:

Scalability and Real-Time Adaptation: When working with enormous datasets that contain millions of users and products, scalability problems persist even if clustering is used to minimize complexity. Another constant problem is adapting in real-time to changing user behaviours.

Cluster Quality: The quality of the clusters determines how well the clustering stage works. Inaccurate suggestions when selecting the clustering algorithm may result from poorly defined clusters, underscoring the need for more effective ways to assess and enhance clustering methodologies.

Complexity of Policy Learning: As clusters change over time, it becomes increasingly difficult to determine the best policies for each one. More research is needed in creating more effective RL algorithms that can manage dynamic clusters with less processing.

A comparison table summarizing work on existing clustering and reinforcement learning-based recommender systems is presented in table 1.

Research objectives:

We carried out work with the following research objectives (RO):

RO1. Apply eight clustering algorithms (K-Means, Hierarchical, DBSCAN, Mean-Shift, GMM, Spectral, OPTICS, BIRCH) on user- and product-side behavioral features, and select the best-performing algorithm using standard deviation as a measure of cluster quality.

RO2. Formulate a Jaccard-similarity-based reward function that maps cluster overlap directly onto Q-learning rewards and use it to drive state-to-state transitions in a 6×6 grid-world MDP.

RO3. Evaluate the resulting recommender on two structurally different public datasets (Amazon, MovieLens) using both RL-internal diagnostics (start-state distribution, states visited, return, steps-per-episode) and standard recommendation-quality metrics (Precision, Recall, F-measure, Hit Ratio) over a sampled user population.

RO4. Characterize where the clustering + RL approach is limited by data sparsity (Amazon: median 1 interaction/user) versus where it benefits from denser interaction histories (MovieLens: mean ~149 ratings/user).

Table 1. A comparison of existing clustering and reinforcement learning based systems

Study	Clustering	RL Method	State Definition	Reward Basis	Dataset(s)	Reported Metrics
[30]	Bi-clustering	Q-learning	Bicluster	Collaborative signal	MovieLens	Not standard metrics reported
[4]	Bi-clustering	MDP / Q-learning	Bicluster	Rating-based	MovieLens, others	Precision, Recall

[28]	Bi-clustering	Q-learning	Bicluster	Rating-based	Public RS datasets	Precision, Recall, coverage
[20]	—	Self-supervised RL	Sequential	Supervised signal	Public RS datasets	Recall, NDCG
[29]	Graph-based	RL + GNN	Graph node	Graph-derived	Public RS datasets	Standard ranking metrics
This work	**K-Means (8 algorithms applied)**	**Q-learning, 6×6 grid**	**Cluster ↔ grid cell**	**Jaccard similarity**	**Amazon, MovieLens**	**RL diagnostics metrics**

Methodology:

The three primary components of our proposed method are:

Clustering: This process simplifies and lowers the complexity of the data by grouping users and products with comparable interaction patterns. The system can operate with a smaller, more effective state space by grouping the data into manageable clusters, which increases the system's overall scalability.

Reinforcement Learning: Following clustering, a reinforcement learning agent explores and utilizes the clusters to produce dynamic and tailored recommendations. The agent continuously improves the suggestions it makes by adapting in response to user interactions.

System Integration: The system effectively manages big datasets by combining RL with clustering, which lowers computing load and improves the recommendation process's flexibility and customisation. The system will be able to adapt to changing user preferences and grow with the amount of data through this system integration.

Tools, Technologies, and Techniques:

Programming language: Python, the primary language used for data analysis, machine learning, and reinforcement learning, is used due to its extensive libraries, and it is user friendly.

Libraries for Data Processing, Clustering Algorithms and Reinforcement Learning: Several libraries such as:

NumPy – used for numerical operations and handling multi-dimensional arrays and matrices.

Pandas – it is ideal for data manipulation and preprocessing. It helps in processing large datasets and make it easier to structure in user-item interaction.

Scikit-learn – it provides the implementation of all clustering algorithms which are K-means, DBSCAN, Hierarchical, Mean shift, Gaussian mixture, Spectral, OPTICS and BIRCH. It is used to perform clustering, evaluate the quality of cluster, and optimize the hyperparameters.

Matplotlib – it is used for plotting graphs and visualize the distribution of clusters and the agent's learning progress.

Seaborn – it is an advanced visualization toolkit based on Matplotlib that makes data plotting simpler and produces more visually appealing charts.

Statistics – it provides the function to perform statistical calculations such as finding mean, median, mode and standard deviation.

Open AI Gymnasium – it is a popular library for creating and managing custom environments of RL training agents. It specifies the framework, including states, rewards, policies, and actions that are used for handling interaction with the RL model.

PyTorch – it is a popular tool for building machine learning and AI models, especially in deep learning and reinforcement learning. It offers flexibility, efficiency, and powerful features, making it ideal for modern RL tasks and implementing reinforcement learning algorithms.

Clustering Techniques: Selecting the best clustering algorithm is a critical part of the research methodology, as it groups users and products according to their interaction pattern. By organizing them into manageable clusters, it enables the RL agent to operate within a desired state space, thus, to improve efficiency and scalability. The eight clustering algorithms

used for this paper are: K-means partitional Clustering, Hierarchical Clustering, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Mean Shift Clustering, Gaussian Mixture Model, Spectral Clustering, Ordering Points To Identify the Clustering Structure (OPTICS) and Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH).

Reinforcement Learning Techniques: A model free RL algorithm, Q-Learning, is used to learn the optimal policy by updating the Q- table based on the rewards. It is ideal for discrete action spaces like recommending products from different clusters [31].

Techniques for Integrating Clustering and Reinforcement Learning: The integration of clustering and RL improves the system ability to provide scalable, accurate and customized recommendations. The key technique used to combine them in context to product recommender system are:

Dimensionality Reduction: In recommender systems, clustering is a technique that groups similar users and items into clusters to reduce dimensionality. As a result, exploring each state separately would take less time. Product clusters with a single "user cluster" are produced by clustering algorithms, which find patterns in how users interact with products. By focusing on groups of comparable users and items, the RL agent can make informed decisions and converge to the best recommendation techniques more quickly. This lowers the complexity of the state space.

State-Action Mapping: In Reinforcement Learning (RL), states and actions are used to reflect the environment. The agent moves between states and performs actions that cause transitions; each action has a reward associated with it. Clusters are groups of individuals or items with comparable attributes that the agent maps to states. The agent then acts as it moves across clusters with the goal of suggesting products from one cluster to the next. Clustering makes the state-action space simpler, which improves the efficiency and manageability of the learning process.

Reward Function Design: Reinforcement Learning (RL) relies on a reward function to guide agents in making the best decisions. As K-Means being a hard clustering algorithm, assigns every user to exactly one cluster, so our reward function is based on items overlapping across the clusters, instead of users overlapping across the clusters. Reward function determines the likelihood of choice of a user for a product. High rewards are given to items from clusters that closely match with other items in the clusters, while low rewards are given to items from less overlapping clusters [30]. To calculate the reward, a renowned similarity metric, Jaccard similarity is employed. Mathematically formulation of Reward function is given in Eqn. (1).

$$R(s,s') = J(I(s),I(s')) = \frac{|I(s) \cap I(s')|}{|I(s) \cup I(s')|}, R(s,s') \in [0,1] \quad (1)$$

In Eqn. (1), s denotes current state/cluster and s' denotes next state/cluster. Reward function $R(s,s')$ effectively measures overlap count of items I in s and s'.

Q-Learning:

Q-Learning is a fundamental reinforcement learning algorithm used to optimize action-selection policies in an agent's environment, aiming to maximize cumulative rewards over time. Q-learning is an off-policy, model-free algorithm that learns the optimal policy independently of the current policy. It works by initializing a Q-table, which stores Q-values for state-action pairs $Q(s,a)$. Each episode involves observing the current state, selecting an action, and observing the reward. The Q-value is updated using the Q-learning update rule, with a learning rate and discount factor controlling how much new information overrides old information. The next state is set as the current state, and the episode is terminated if the goal state is reached or after a predefined number of steps [32].

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma amaxQ(s',a) - Q(s,a)) \quad (2)$$

Where,

α = Learning rate
r = Reward
γ= Discount rate

In the context of product recommender systems, Q-learning can be used to dynamically adjust recommendations based on user interactions, leading to personalized and effective recommendations over time. Despite its simplicity and model-free nature, Q-learning faces challenges in scaling to large state and action spaces, but it remains a powerful tool for many reinforcement learning applications.

Data Collection and Preprocessing:

Amazon Dataset: This dataset includes 1462 products with attributes such as category, discounted price, ratings, products, and users. It provides a broad view of e-commerce items, helping to understand user preferences and product characteristics.

Movie Lens Dataset [33]**:** This dataset contains 100,004 entries (Products) with movies of various genres and 671 users. This dataset is valuable for understanding user preferences in entertainment and can be used to draw parallels in recommendation logic.

The data is cleaned to remove any duplicates, irrelevant entries, or erroneous data points.
Features such as price or ratings are normalized to a standard scale.
Categorical data (e.g., product description or users) is encoded using one-hot encoding to convert them into numerical formats suitable for machine learning algorithms.
To ensure data is formatted consistently, we replaced spaces with commas in users and product IDs, so that clusters are well defined.

Implementation of Clustering Algorithms:

Clustering serves as the initial phase to segment users or products into groups based on similarities, which can then be used to make more targeted and personalized recommendations. Instead of dealing with millions of individual user-product interactions, clustering reduces the problem by making smaller/predefined number of groups which significantly improves the computational efficiency without compromising on the performance. Eight Clustering algorithms are chosen to cover a variety of clustering approaches for the analysis.

Each clustering algorithm is implemented using a consistent coding structure to ensure comparability. The only change in the code across implementations was the specific clustering function called. It represents a distinct method for identifying clusters.
The key concepts of our proposed method and related code are:

The main goal of the code is to group users and products into predefined number of clusters i.e. 36 clusters, which simplifies the recommendation process by reducing the number of unique interactions. This process generates a representation of users/products which share the similar behaviors.

A dictionary object named clusters is created to group users by their cluster label and another dictionary object Dictobj is created to group product IDs and users associated with each product cluster. This is important to understand which users are interacting with which types of products.

To ensure data integrity and highlights the issues with the clustering algorithms, code is generated which prints any duplicate values (users that appears more than once in the clusters) and missing values (users which are intended to be in the cluster but are missing) in the cluster.

A list and set operations are also used to analyses and verify that the system is correctly assigning the users and products to the clusters without redundancy and gaps.

The information about each cluster, which contains both users and products is printed which allows to see the relationship between users and products in each cluster and it defines

the user-product interaction. This will help system to better understand the patterns and similarities in user behavior and product interactions.

Clustering techniques are used to group subsets of users and products with similar interaction patterns. Clustrows(users) and Clustcolums(products) are generated which provide a structured way to link the specified users and products. This makes easy for the recommender system to focus on users and products for personalized recommendations. Instead of dealing with all the users and products individually, grouping them into clusters reduces the overall complexity of the recommendation problem.

The biclusters serve as a foundation for defining the states in the reinforcement learning environment, each clusters represents a state in the RL model and actions are taken by the RL agent, moving between clusters, or recommending products from a specific Clustcolums to users in the corresponding Clustrows.

While selecting the best clustering algorithm, standard deviation is also considered to find the consistency and reliability of the clustering algorithm. The smaller the value of standard deviation, greater will be the reliability of the algorithm.

Based on the outlined factors, K-means clustering has been identified as the best fit for integration with reinforcement learning in this recommender system. Below is a detailed explanation of the rationale, methodology and advantages of using K-means clustering:

Why is K-means Selected?

K-means is a centroid-based clustering algorithm that iteratively assigns data points to predefined number of clusters, adjusting cluster centroids to minimize within-cluster variance. We generated 36 clusters to best fit the size of 6×6 2D grid world environment as shown in figure 2.

K-means is computationally efficient, even for large datasets, making it ideal for the scale of a product recommender system.

It satisfies the data integrity as there are no missing values, all the users are assigned to the clusters, thus producing zero overlap and good separation, making it an ideal, accurate, and robust choice for RL.

It has relatively low standard deviation compared to other algorithms, thus providing consistent clustering results. It indicates high consistency and reliability.

As our objective is to measure how many similar products are in a cluster rather than determining the shape of the clusters with strong dependence on distance-to-centroid, so, we used standard deviation (SD). As SD solely measures how data points are spread out within a single group. Silhouette and Davies–Bouldin Index assumes clusters are convex or spherical, whereas SD measures the exact spread along the axes without assuming cluster geometry. Davies–Bouldin Index best suites to density-based algorithms like DBSACN but is not suitable for centroid based algorithms. Moreover, computing Silhouette, Davies–Bouldin Index and Calinski–Harabasz Index is slow on massive datasets whereas measuring SD is computationally cheaper.

It provides compact and easy to interpret clusters which can be used for reinforcement learning. When implementing the code, each cluster serves as a basis for RL-based recommendations by containing users and products with comparable interaction patterns.

Integration of Reinforcement Learning:

Integrating clustering with Reinforcement learning in a product recommender system establishes a sophisticated code of framework. Clustering is used to focus on user-product interactions, grouping subsets of users and products according to common patterns of interaction. By considering not just the individual attributes of users but also the relationship between them and products, this method goes beyond conventional clustering. Clusters were created in this study by analysing which users interacted with which products and classifying them into user-product pairings that showed robust, reciprocal behavioural patterns. This

approach was successful in lowering the recommendation problem's overall complexity. By using predetermined groupings of users and products, the cluster-based method enabled targeted recommendations rather than suggesting things for all users and all items. This made the recommendations much more relevant because it enabled the system to concentrate on highly specific user-product combinations. In addition to increasing the system's computing efficiency, clustering made sure that the quality of recommendations was maintained as the system grew to handle more datasets by reducing the range of interactions. Ultimately, Clustering's worth in the recommendation framework was demonstrated by the way it was integrated into the recommendation framework, which offers an influential way to refine the recommendations and give users more relevant and customized choices [34]. Structurally our proposed method is presented in figure 1. Figure 1 shows that at a high-level proposed work is broken into four components i.e. Primary components, Tools and Technologies, Integration Techniques and Data Processing. Each component is further sub-divided into sub-components.

To minimize the complexity and establish organized states, RL leverages the strengths of clustering. At a finer resolution, clustering is used to study user-product interactions. The integration guarantees that the system can optimize recommendations using RL approaches while dynamically learning and adapting to user actions.

The steps of integrating RL with clustering are:

The extracted clusters (from K-means clustering) are provided to the RL framework where states represent the clusters, containing the subsets of users and products and rewards are the evaluation based on the success of recommendations within clusters.

A 2D grid (6x6) environment as shown in figure 2, is used to represent a grid world with defined actions as Up = 0, Down = 1, Left = 2, Right = 3. Start and terminal states are determined dynamically based on user interactions and recommendations.

Other commands used to run the RL agent in a 2D grid are:

Step(action, env) = it executes an action, make transition of the agent to the next stage and calculates the rewards.

Reset(env, u1I, u1R) = resets the environment to an initial state based on user-item data.

Render(env) = it displays the current stage of the grid.

Close() = it cleans up the environment

The reinforcement learning logic implements the Q-learning to train the agent by learning Policy, which extracts the optimal sequence of actions leading to a goal state. An optimal policy is then extracted from the build Q-table. Extracted policy is then applied to move on the environment reaching to a goal state and obtaining personalized item recommendations.

Cross validation, particularly 5-fold cross validation is used to ensure the reliability of model performance. By testing the model on multiple validation sets, cross validation ensures that the performance metrics (e.g., precision, recall, return) are not biased by the specific data used in a single train-test split. It aids in evaluating the RL model's ability to generalize to new data, by decreasing the chance of overfitting.

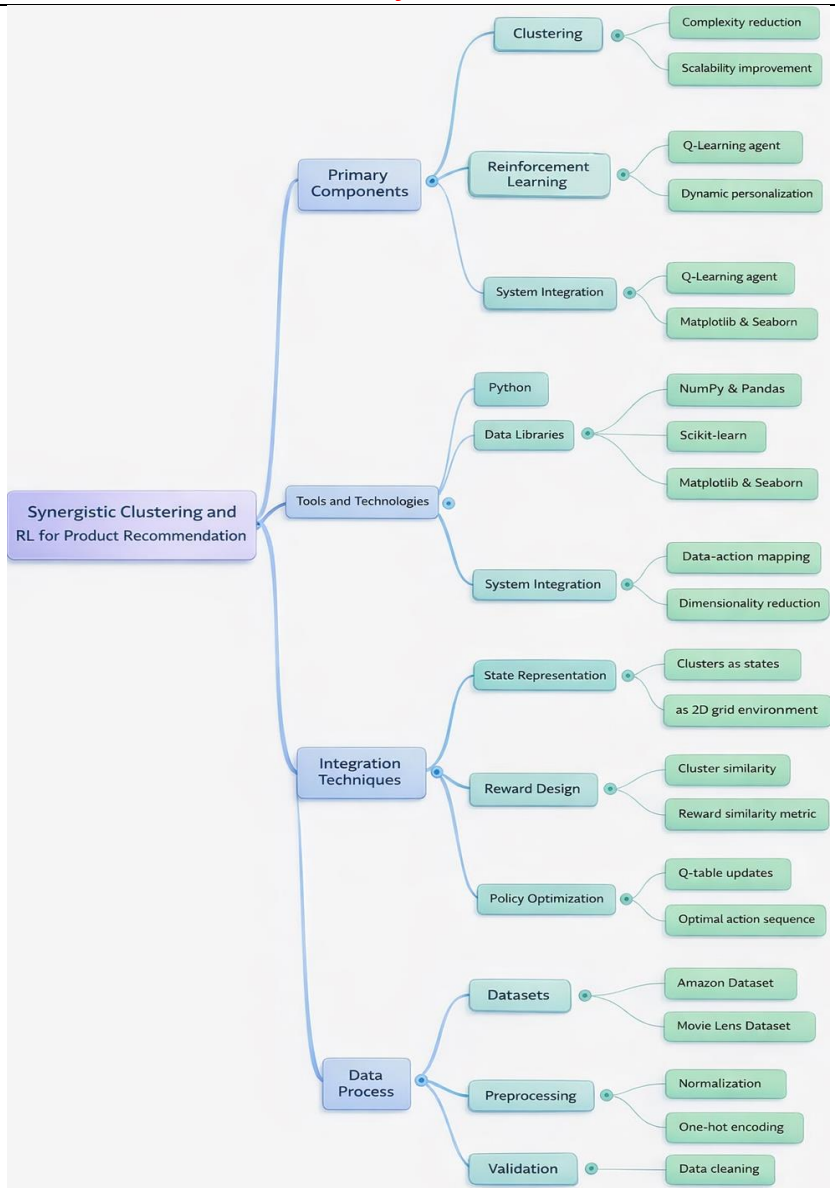


Figure 1. Synergistic Clustering and RL for Product Recommendation

	0	1	2	3	4	5
0	C_1	C_2	C_6	C_7	C_{15}	C_{16}
1	C_3	C_5	C_8	C_{14}	C_{17}	C_{26}
2	C_4	C_9	C_{13}	C_{18}	C_{25}	C_{27}
3	C_{10}	C_{12}	C_{19}	C_{24}	C_{28}	C_{33}
4	C_{11}	C_{20}	C_{23}	C_{29}	C_{32}	C_{34}
5	C_{21}	C_{22}	C_{30}	C_{31}	C_{35}	C_{36}

Figure 2. 6×6 2D grid world environment

Challenges faced and how they were addressed:

Challenges Faced	Solution
Data Sparsity – User-product interaction matrix often has missing values, leading to sparse matrix which is difficult to find	K-means Clustering is used to group users and products in meaningful clusters, as it is giving the minimum number of missing values of users and

correct relationship and generate recommendations in RL.	products, the system was able to concentrate on significant clusters rather than the sparse dataset.
	K-means Clustering is done to reduce the sparsity by identifying the subsets of users and products with significant overlap.
Scalability - For big datasets clustering algorithms become intensive, taking ages to run, and slowing down the system.	K-means clustering was selected due to its high quality and computational efficiency, and it was optimized with Python parallel processing. Clustering reduced the problem space further by dividing data into smaller subsets and utilizing modularized data processing helped to reduce complexity.
Reward Function Design - Designing a reward function that accurately represents user satisfaction was complex	Rewards were based on user satisfaction metrics like Jaccard similarity, refined through iterations.
Cold Start Problem - The lack of historical interaction data made it challenging to promote new products or to suggest products to new consumers.	Clusters helped mitigate this issue by associating users and products in similar existing clusters

Results & Analysis:

This section provides the results of integrating clustering with reinforcement learning (RL) for our proposed product recommendation method. The evaluation focuses on user-product interactions, clustering effectiveness, and the performance of the RL agent in optimizing recommendations. We applied the following four criteria to measure the performance for our proposed method.

How many times a state is selected as the start state.

How many states are visited by each user to obtain recommendations?

How much Returned earned to reach a goal state?

As RL agent works in episodes to learn about the environment, so it is important to determine how many steps are taken by the agent in each episode?

This section provides an in-depth presentation and critical analysis of the results obtained from the reinforcement learning (RL)-based recommender system. It delves into key performance metrics, offering a comprehensive evaluation of the system's ability to deliver personalized recommendations effectively. Furthermore, the discussion highlights the insights gained from integrating clustering techniques with reinforcement learning, emphasizing how the combination enhances the model's capability to adapt to user preferences and improve overall performance. By exploring these aspects, this section sheds light on the practical implications and potential improvements of employing such a hybrid approach in recommendation systems.

Presentation of Findings:

Selection of Suitable Clustering Algorithm:

Ensuring the integrity of the findings is a crucial component of any clustering process. Redundancy and missing variables were effectively avoided during the clustering process, ensuring that no user was left without a cluster allocation and that no user assignments overlapped. This was accomplished by thoroughly preparing the data, removing any unnecessary or inaccurate entries, and carrying out exacting verifications to guarantee that every user and product was assigned appropriately. A more balanced representation of users and products across the clusters was ensured by the approach, which also showed that the clusters were well-distributed and populated without duplication. By calculating the standard

deviation of the cluster assignments, which revealed minimal variance, the algorithm's performance was evaluated. The K-means algorithm's dependability and robustness in segmenting the dataset into uniform groups was demonstrated by the clusters' high consistency, as indicated by its low standard deviation on both datasets. These reliable and distinct clusters served as stable states from which the agent can make significant judgments during the reinforcement learning phase that followed. DBSCAN and OPTICS have low standard deviation, but they didn't perform well in clustering formation that is why, we selected the K-means clustering algorithm due to it's low standard deviation. Figure 3 shows standard deviation of applied clustering algorithms on Amazon dataset, while figure 4 presents this analysis for Movie Lens dataset.

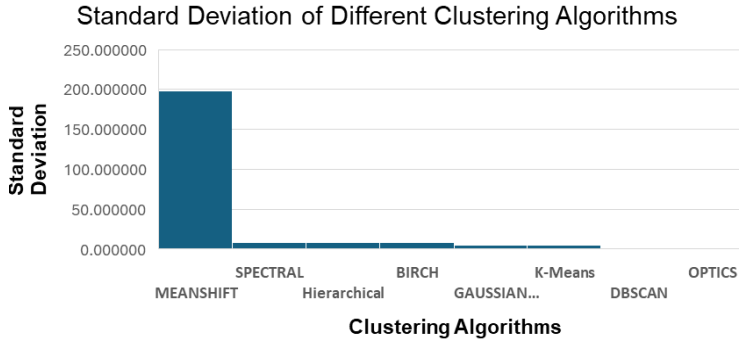


Figure 3. Standard deviation of different clustering algorithms on Amazon dataset

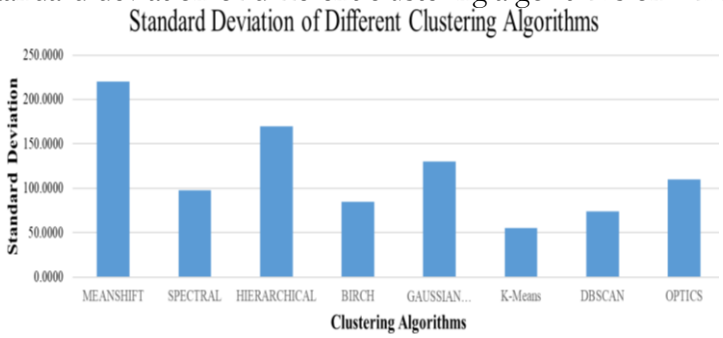


Figure 4. Standard deviation of different clustering algorithms on Movie Lens dataset

Clustering helped us to reduce the computational search space, to identify appropriate items by focusing on products that are part of the same cluster (Cluster), which results in simpler suggestions. Additionally, because the clusters may be easily mapped to states in an RL framework, clustering supports the incorporation of Reinforcement Learning (RL). Designing transitions between clusters is made easier by this mapping, which also aids in defining practical tactics with quantifiable benefits. The system's scalability is enhanced by clustering, particularly when working with big datasets. Users and goods are grouped together to simplify user-product interactions and improve recommendation efficiency while preserving tailored and focused suggestions.

K-Means Cluster Analysis:

After selecting K-Means as the best clustering algorithm on the bases of lowest SD, we analyze the generated clusters of K-Means in this section. The K-means algorithm played a central role in the success of our proposed system by effectively grouping users and products into 36 distinct clusters. Each cluster represents a subset of users or products exhibiting similar behaviors and interactions. The selection of 36 clusters was a critical decision aimed at balancing the complexity of the problem with computational efficiency. By partitioning the data into these manageable groups, we reduced the vast number of user-product interactions into smaller, more focused sets, which made the recommendation process significantly faster and more targeted. Users within the same cluster share similar preferences or behaviors, while

products in the same cluster tend to attract similar user groups. This clustering not only helps in identifying patterns but also improves the system's ability to recommend products that are more relevant to users based on their collective characteristics. The K-means algorithm's ability to converge on meaningful, compact clusters also helped us to ensure that the clustering process was both efficient and effective, with the user-product interactions being more predictable within these defined groups. Figure 5 presents the distribution of users across 36 clusters for the Amazon dataset. On average number of users in each cluster is 40, whereas a maximum of 70 users is contained in the 13th cluster. Figure 6 presents production distribution across 36 clusters.

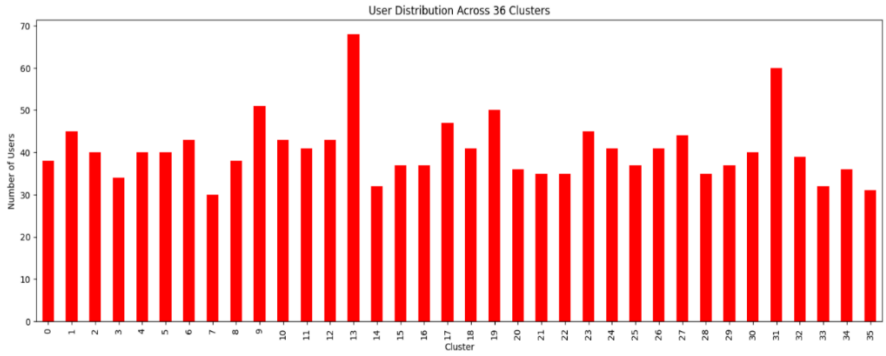


Figure 5. K-Means User Distribution Across 36 clusters on Amazon dataset

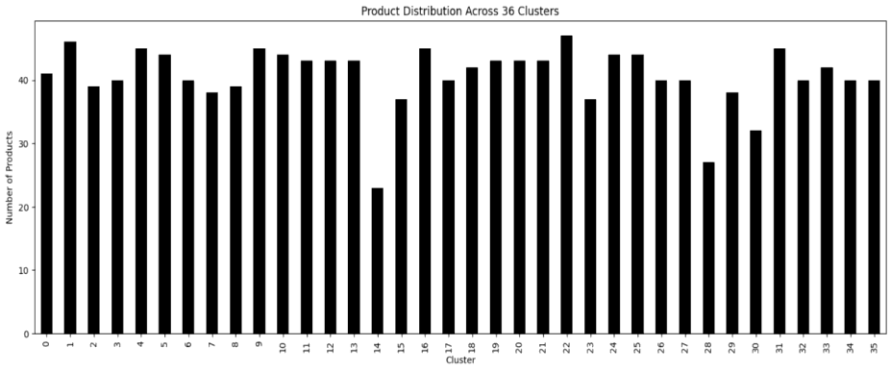


Figure 6. K-Means Product Distribution Across 36 clusters on Amazon dataset

Figure 7 shows user's distribution across 36 clusters of Movie Lens dataset. Although each cluster contains variable number of users, it is observed that a minimum of ten users is at least contained in each cluster. A maximum of twenty-five users is contained in clusters 1 and 18.

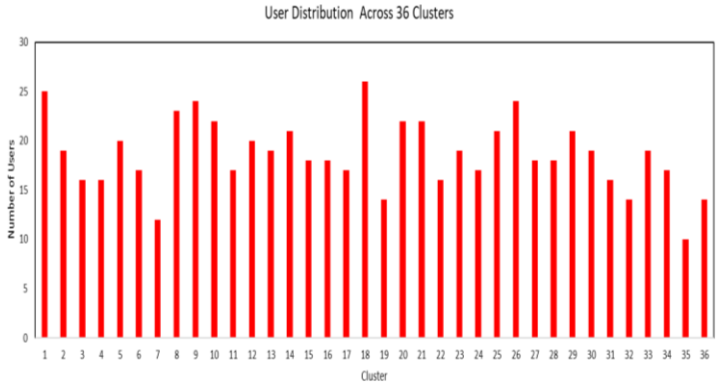


Figure 7. K-Means User Distribution Across 36 clusters on Movie Lens dataset

Figure 8 shows distribution of products/ movies across 36 clusters for Movie Lens dataset. Except cluster number 19, every cluster is encompassed by approximately 200 to 500 products/ movies. Cluster number 19 contains approximately 2200 products/ movies.

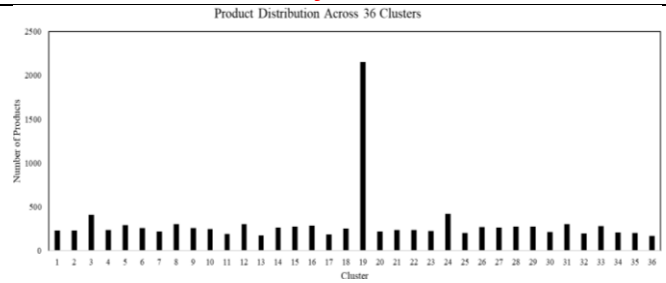


Figure 8. K-Means Product Distribution Across 36 clusters on Movie Lens dataset

Reinforcement Learning Results

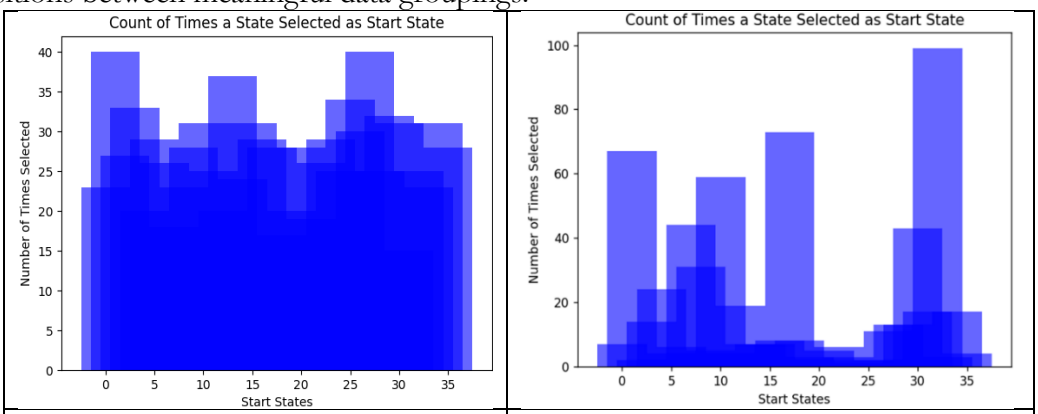
The integration of Q-learning with a grid-world environment to optimize recommendations in a personalized product recommender system is defined below:

Q-learning Performance:

The Q-learning process in the 6x6 grid world environment involves training an RL agent to navigate states, defined by clusters, and optimize its actions to maximize cumulative rewards. Through this structured training, the Q-learning algorithm enables the agent to develop an optimal policy for transitioning between states, with the goal of maximizing rewards while navigating the grid environment effectively. Here's a detailed explanation of the process:

Actions: The agent can move in four possible directions— Up = 0, Down = 1, Left = 2, or Right = 3 within the grid. These movements allow the agent to transition between states.

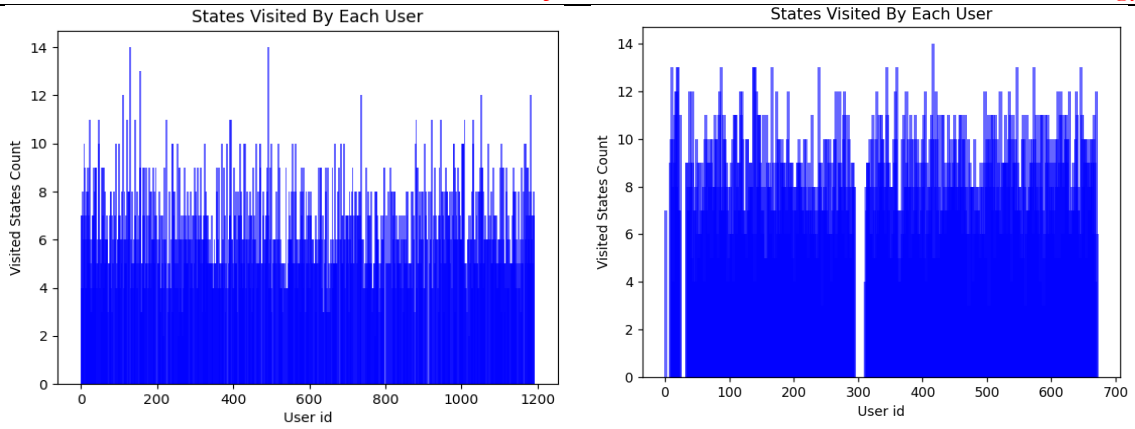
States: Each state in the grid is associated with a cluster generated during the clustering phase. These clusters represent groups of related data points, and the states dynamically reflect their structure. This mapping ensures that the agent's movements within the grid correspond to transitions between meaningful data groupings.



(a) Amazon (b) Movie Lens

Figure 9. Count of Times a State Selected as Start State on Amazon dataset

The most frequently selected state reaches a count of about 40 on the Amazon dataset, figure 9-a, while others fall slightly below. The most frequently selected state on the MovieLens dataset reached a maximum count of 100, figure 9-b. A few states are selected a very small number of times. The overlap shows the repetition of experiments on the start states. A balanced distribution indicates that the Q-learning agent explored a wide range of starting states.

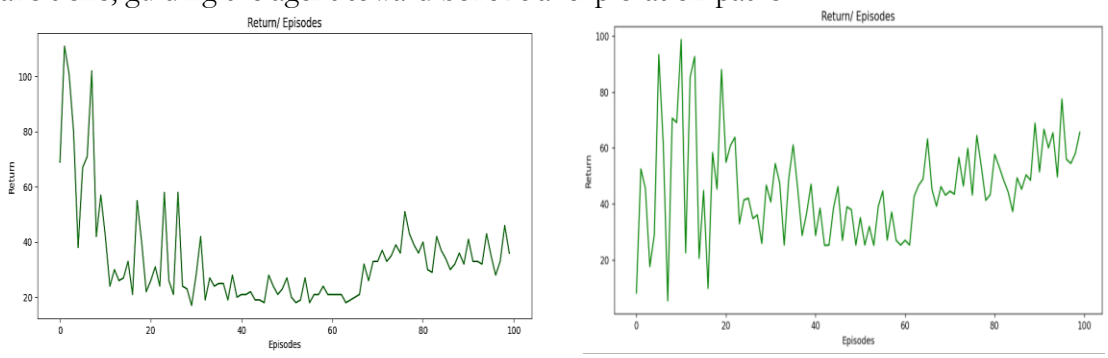


(a) Amazon **(b) Movie Lens**

Figure 10. Distribution of number of states visited by each user

The graphs in figure 10, can be used to assess the diversity of states visited to yield recommendations for each within the 6x6 grid environment, which then impact the effectiveness of learned policies.

Rewards/Returns: Rewards are used to assess the value of transitioning between states. Our reward function considers the similarity or dissimilarity between clusters and uses this contextual information to assign reward values. Higher rewards indicate more effective transitions, guiding the agent toward beneficial exploration paths.

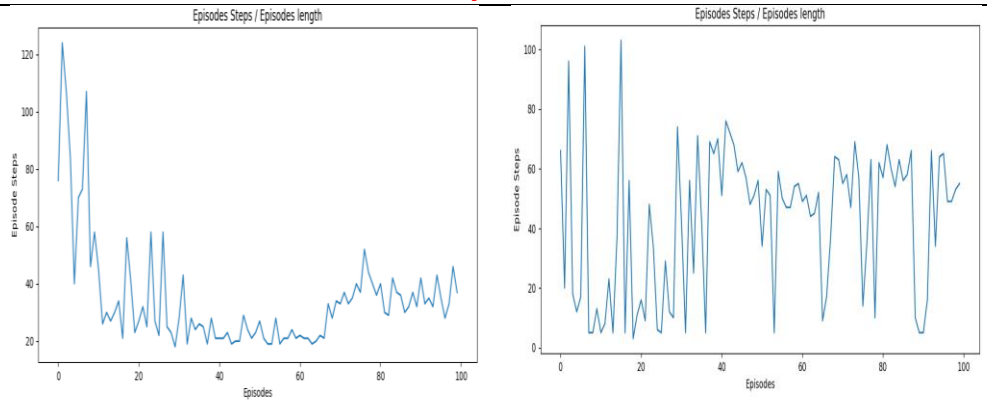


(a) Amazon **(b) Movie Lens**

Figure 11. Return earned in each episode for a single user

The figure demonstrates the trend of return values across 100 episodes in a 6×6 grid-world environment. For Figure 11(a), the Amazon dataset, during episodes 1 to 15, the agent exhibits significant variability and achieves the highest return values, reflecting exploratory behavior. As episodes progress, the return values stabilize, indicating the agent's improved policy and convergence toward optimal decision-making. The gradual upward trend after episode 50 suggests consistent learning, where the agent achieves more efficient navigation through the environment. Almost same behavior is observed on Movie Lens dataset for RL agent in figure 11-b.

Episodes: The agent undergoes training for 100 episodes, with each episode representing a complete learning cycle. At the start of each episode, the agent is placed in an initial state derived from Jaccard similarity—a measure of set similarity between users. This initialization ensures that the agent begins its exploration from meaningful and relevant states, improving the learning process's efficiency.



(a) **Amazon** (b) **Movie Lens**

Figure 12. Number of steps taken for each episode for a single user.

The relationship between the number of steps taken by the agent in a reinforcement learning environment and the number of episodes is depicted in Figure 12(a) for the Amazon dataset and Figure 12(b) for the MovieLens dataset. High unpredictability and a greater number of steps are characteristics of early episodes (about episodes 1–20), which represent the exploratory stage in which the agent is learning the dynamics of the environment. The number of steps steadily decreases from episodes 20 to 50 as training progresses, suggesting that the agent is discovering a more effective way to accomplish its goals. Steps gradually grow beyond episode 50, though, perhaps because of the agent trying out different tactics or running into increasingly challenging situations, which causes a little performance instability. As the agent progresses through the training process, the graphic generally depicts its learning curve.

Practical Implications and Deployment Considerations:

Deploying the proposed method in a real-world scenario may raise three considerations beyond the offline evaluation reported here. First, cluster re-computation cost: because states are defined by K-Means cluster membership, catalog or user-base growth requires periodic re-clustering and Q-table retraining; the current pipeline retrains from scratch rather than incrementally updating existing clusters, which would not scale to real-time catalog changes without further engineering. Second, cold start: users or items with no cluster overlap under the Jaccard start-state rule, currently receive no recommendation at all rather than a fallback (e.g., popularity-based) recommendation. Third, the reward function's reliance on item-set overlap means recommendation quality is bounded by how well clusters were formed upstream; on sparse interaction data (as in the Amazon case) this is the dominant practical bottleneck.

Conclusion and Future Work:

In this work we combined conventional CF based recommendation technique with RL. Previously, CF was used as a standard standalone technique to generate recommendations and was found to face failure to evolve with changing user rating preferences. Specifically, we applied eight clustering algorithms on two standard datasets (In line with RO1 and RO4 of research objectives). Then each clustering algorithm's generated clusters are evaluated for their quality and standard deviation is used as a quality measure. On both datasets, K-Means clustering algorithm yielded minimum standard deviation. So, we selected K-Means clusters for our RL part. For RL part each cluster is mapped to a 6×6 grid world environment in a diagonal manner. RL agent is designed to move on this environment to get recommendations. RL agent started it's movement from a determined start state and then took actions that moved agent to the states that give item recommendations that best matches with user rating preferences. For any change in user rating preferences agent will change its actions and paths to incorporate these changing preferences. So, in a nutshell our proposed method is able to utilize the benefits of both clustering and RL. In future this work can be extended to more

datasets to verify its robustness. Moreover, we used only Q-Learning in this work, more robust RL methods like Deep Q-networks and Actor-Critic methods can be used to further enhance the performance of the proposed method. Another possible future direction can be a tree-based environment instead of a grid-based environment. Current work utilizes RL diagnostic metrics like states visited and return earned; in future we intend to expand this work to include GPU Acceleration done by other researchers.

Acknowledgment:

All authors contributed equally. Moreover, manuscript has not been published previously nor submitted to any other journal currently.

References:

[1] M. Ayub, M. A. Ghazanfar, M. Maqsood, and A. Saleem, "A Jaccard base similarity measure to improve performance of CF based recommender systems," *Int. Conf. Inf. Netw.*, vol. 2018-January, pp. 1–6, Apr. 2018, doi: 10.1109/ICOIN.2018.8343073.

[2] T. S. Madhulatha, "An Overview on Clustering Methods," *IOSR J. Eng.*, vol. 02, no. 04, pp. 719–725, May 2012, doi: 10.9790/3021-0204719725.

[3] K. Sivamayil, E. Rajasekar, B. Aljafari, S. Nikolovski, S. Vairavasundaram, and I. Vairavasundaram, "A Systematic Study on Reinforcement Learning Based Applications," *Energies 2023, Vol. 16, Page 1512*, vol. 16, no. 3, p. 1512, Feb. 2023, doi: 10.3390/EN16031512.

[4] A. Iftikhar, M. A. Ghazanfar, M. Ayub, S. Ali Alahmari, N. Qazi, and J. Wall, "A reinforcement learning recommender system using bi-clustering and Markov Decision Process," *Expert Syst. Appl.*, vol. 237, p. 121541, Mar. 2024, doi: 10.1016/J.ESWA.2023.121541.

[5] H. Yin, A. Aryani, S. Petrie, A. Nambissan, A. Astudillo, and S. Cao, "A rapid review of clustering algorithms," *Array*, vol. 30, Jul. 2026, doi: 10.1016/j.array.2026.100904.

[6] M. Bin Tariq and H. A. Habib, "A Reinforcement Learning Based RecommendationSystem to Improve Performance of Students in Outcome Based Education Model," *IEEE Access*, vol. 12, pp. 36586–36605, 2024, doi: 10.1109/ACCESS.2024.3370852.

[7] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval, SIGIR 1999*, pp. 230–237, 1999, doi: 10.1145/312624.312682.

[8] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4321 LNCS, pp. 325–341, Jan. 2007, doi: 10.1007/978-3-540-72079-9_10/Save-Research.

[9] V. Vellaichamy and V. Kalimuthu, "Hybrid collaborative movie recommender system using clustering and bat optimization," *Int. J. Intell. Eng. Syst.*, vol. 10, no. 5, pp. 38–47, 2017, doi: 10.22266/IJIES2017.1031.05.

[10] W. Hassan, S. E. Hosseini, and S. Pervez, "Real-Time Anomaly Detection in Network Traffic Using Graph Neural Networks and Random Forest," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 14542 LNCS, pp. 194–207, 2024, doi: 10.1007/978-3-031-60994-7_16/SAVE-RESEARCH.

[11] R. Ghous, S. E. Hosseini, and S. P. Chattha, "Lung Cancer Survival Period Prediction: Exploring Machine Learning Approaches," *Int. J. Online Biomed. Eng.*, vol. 21, no. 04, pp. 45–60, Mar. 2025, doi: 10.3991/IJOE.V21I04.52889.

[12] GanganiParth, AlyaseriSana, and HosseiniSeyed, "Machine Learning Approaches for Predicting Diabetes Onset: A Comparative Study of XGBoost, Random Forest, and Traditional Models," Mar. 2025, doi: 10.36227/TECHRXIV.174317792.24675569/V1.

[13] S. Mansur *et al.*, "Sales forecasting for retail stores using hybrid neural networks and

sales-affecting variables," *PeerJ Comput. Sci.*, vol. 11, p. e3058, 2025, doi: 10.7717/PEERJ-CS.3058/SUPP-4.

[14] J. Wilson, S. E. Hosseini, and S. Pervez, "Identification of Fake News in Social Media Using Sentimental Analysis," *IEACon 2023 - 2023 IEEE Ind. Electron. Appl. Conf.*, pp. 220–224, 2023, doi: 10.1109/IEACON57683.2023.10370300.

[15] L. Acosta, S. E. Hosseini, and S. Pervez, "Ethical Challenges Associated with Security Vulnerabilities and Data Privacy in Social Networking," *Proc. - Int. Conf. Dev. eSystems Eng. DeSE*, pp. 647–652, 2023, doi: 10.1109/DESE60595.2023.10469464.

[16] M. Ali, S. E. Hosseini, and S. Pervez, "Assessment and Enhancement of Real-Time Recognition of Sign Language Alphabets Through Diverse Machine Learning Techniques," *Proc. - 2025 8th Int. Conf. Data Sci. Mach. Learn. Appl. CDMA 2025*, pp. 85–90, 2025, doi: 10.1109/CDMA61895.2025.00020.

[17] M. Ali, S. E. Hosseini, S. Pervez, and M. Ahmad, "Real-Time Recognition of NZ Sign Language Alphabets by Optimal Use of Machine Learning," *Bioeng. 2025, Vol. 12, Page 1068*, vol. 12, no. 10, p. 1068, Sep. 2025, doi: 10.3390/bioengineering12101068.

[18] R. Ahuja, A. Solanki, and A. Nayyar, "Movie recommender system using k-means clustering and k-nearest neighbor," *Proc. 9th Int. Conf. Cloud Comput. Data Sci. Eng. Conflu. 2019*, pp. 263–268, Jan. 2019, doi: 10.1109/Confluence.2019.8776969.

[19] K. Arai and A. R. Barakbah, "Hierarchical K-means: an algorithm for centroids initialization for K-means," 2007.

[20] X. Xin, A. Karatzoglou, I. Arapakis, and J. M. Jose, "Self-Supervised Reinforcement Learning for Recommender Systems," *SIGIR 2020 - Proc. 43rd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 931–940, Jun. 2020, doi: 10.1145/3397271.3401147.

[21] M. Addanki, S. S, D. B. SLAVAKKAM, R. B. Challagundla, and R. Pamula, "Integrating Sentiment Analysis in Book Recommender System by using Rating Prediction and DBSCAN Algorithm with Hybrid Filtering Technique," Jul. 2023, doi: 10.21203/RS.3.RS-3173405/V1.

[22] M. M. Afsar, "Personalized Recommendation Using Reinforcement Learning," 2022, doi: 10.11575/PRISM/39785.

[23] D. M. Shakoor, V. Maihami, and R. Maihami, "A machine learning recommender system based on collaborative filtering using Gaussian mixture model clustering," *Math. Methods Appl. Sci.*, vol. 49, no. 5, pp. 3589–3602, Mar. 2026, doi: 10.1002/MMA.7801.

[24] X. Li, Z. Wang, R. Hu, Q. Zhu, and L. Wang, "Recommendation algorithm based on improved spectral clustering and transfer learning," *Pattern Anal. Appl. 2017 222*, vol. 22, no. 2, pp. 633–647, Nov. 2017, doi: 10.1007/S10044-017-0671-2.

[25] L. Cui, X. Wang, and T. Gu, "A Generic Data Synthesis Framework for Privacy-Preserving Point-of-Interest Recommender Systems," *2023 Res. Adapt. Converg. Syst. RACS 2023*, vol. 1, Aug. 2023, doi: 10.1145/3599957.3606241;Ctype:String:Book.

[26] S. K. Mann and S. Chawla, "A proposed hybrid clustering algorithm using K-means and BIRCH for cluster based cab recommender system (CBCRS)," *Int. J. Inf. Technol. 2022 151*, vol. 15, no. 1, pp. 219–227, Oct. 2022, doi: 10.1007/S41870-022-01113-6.

[27] S. Hassan, M. Ayub, M. Waqar, and T. khan, "Contrasting Impact of Start State on Performance of a Reinforcement Learning Recommender System," *Int. J. Innov. Sci. Technol.*, pp. 565–581, May 2024, doi: 10.33411/IJIST/202462565581.

[28] M. Waqar and M. Ayub, "A personalized reinforcement learning recommendation algorithm using bi-clustering techniques," *PLoS One*, vol. 20, no. 2, p. e0315533, Feb. 2025, doi: 10.1371/JOURNAL.PONE.0315533.

[29] C. Fan and S. Fujita, "Reinforcement Learning-Based Recommender Systems Enhanced With Graph Neural Networks," *IEEE Access*, vol. 13, pp. 150228–150243,

2025, doi: 10.1109/ACCESS.2025.3598092.

[30] S. Choi, H. Ha, U. Hwang, C. Kim, J.-W. Ha, and S. Yoon, "Reinforcement Learning based Recommender System using Biclustering Technique," Jan. 2018, Accessed: May 19, 2024. [Online]. Available: https://arxiv.org/abs/1801.05532v1

[31] J. Wang, A. Karatzoglou, I. Arapakis, and J. M. Jose, "Reinforcement Learning-based Recommender Systems with Large Language Models for State Reward and Action Modeling," *SIGIR 2024 - Proc. 47th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, vol. 1, pp. 375–385, Mar. 2024, doi: 10.1145/3626772.3657767.

[32] S. Amin, "An Introduction to Reinforcement Learning and Its Application in Various Domains," *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/979-8-3693-1738-9.ch001*, pp. 1–25, Jan. 1AD, doi: 10.4018/979-8-3693-1738-9.CH001.

[33] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, Dec. 2015, doi: 10.1145/2827872;Wgroup:String:Acm.

[34] C. Tran, J. Y. Kim, W. Y. Shin, and S. W. Kim, "Clustering-Based Collaborative Filtering Using an Incentivized/Penalized User Model," *IEEE Access*, vol. 7, pp. 62115–62125, 2019, doi: 10.1109/ACCESS.2019.2914556.