ENHANCING RECOMMENDER SYSTEMS WITH DEEP REINFORCEMENT LEARNING AND KNOWLEDGE GRAPH EMBEDDINGS

Neeraj Tiwary [1*] ***, Shahrul Azman Mohd Noah*** [2*]***, Fariza Fauzi*** [3]***, Tan Siok Yee*** [4] ***and Sumaia Al-Ghuribi*** [5]

[1, 2, 3, 4] Faculty of Information Science & Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

[5] Software Engineering Department, Prince Sattam Bin Abdulaziz University, Alkharj 11942, Kingdom of Saudi Arabia

Emails: neeraj.tiwary@gmail.com[1], shahrul@ukm.edu.my[2*], fariza.fauzi@ukm.edu.my[3], esther@ukm.edu.my[4], s.alghuribi@psau.edu.sa[5]

ABSTRACT

Deep Reinforcement Learning (DRL), a subfield of machine learning, has shown remarkable potential in various domains, including recommender systems (RSs). This study leverages DRL to improve RS performance by effectively modeling user preferences and addressing their unique needs. A knowledge graph (KG) is constructed using product information, such as features and historical purchase data, to serve as the environment for the Markov Decision Process (MDP) within the DRL framework. The KG is enriched with embeddings to enable efficient navigation and enhance its utility. The Actor-Critic model in DRL employs these embeddings within the MDP, enabling a more accurate representation of user preferences. Central to this approach is the Representation of User Preferences via Path Embedding Propagation (RUPPEP), which serves as the study's core contribution. Experimental results demonstrate that DRL-based RSs achieve superior performance metrics, with a 13.26% improvement in NDCG for the Amazon Cell Phones dataset and a 15.43% increase for the Amazon Beauty dataset compared to the best SOTA baseline model, highlighting their potential to advance the field of recommendation systems.

Keywords: Recommender systems; Deep reinforcement learning; Knowledge graph; Knowledge graph embedding; Markov decision process.

1.0 INTRODUCTION

Recommender Systems (RSs) have transformed various industries [1], leveraging techniques such as collaborative filtering (CF) [2], matrix factorization [3], and deep learning (DL) [4] to improve customer engagement and drive cross-sell and up-sell opportunities. These approaches have significantly contributed to the success of RSs. However, despite their achievements, these methods often fail to understand user preferences fully and cannot provide the most relevant product recommendations [5]. Given the pivotal role of RSs in shaping user decisions, addressing these limitations is crucial to building user trust and delivering highly personalized recommendations.

Deep Reinforcement Learning (DRL), which integrates the capabilities of DL and reinforcement learning (RL), focuses on training agents to learn from interaction trajectories within a given environment. This paradigm is particularly effective in scenarios requiring adaptive learning from interactions, such as human-robot collaboration, where agents leverage real-time feedback to infer and respond to dynamic user preferences. DRL has demonstrated its potential in various cutting-edge research applications, including strategic decision-making in AlphaGo, complex control in autonomous driving systems, and immersive dynamics in interactive video games. These successes highlight DRL's versatility and its promising role in advancing interactive and adaptive systems across diverse domains [6].

The creation of an adequate environment is critical for the successful application of DRL. Knowledge Graphs (KGs), based on linked data principles, provide a structured and heterogeneous information network that encapsulates entities, their attributes, and relationships. These structured representations are pivotal in enhancing DRL-driven applications, particularly in RSs, where personalization is paramount.
Researchers have explored integrating DRL into RSs to improve recommendation quality and user experience. Notable approaches, such as PGPR [7], ADAC [8], KGDQN [9], EKar [10], ReMR [11], RKGR-RNS [12], HR-RLKG [13], and MES [14], employ RL within a Markov Decision Process (MDP) framework to identify optimal

paths between user-item pairs [15]. However, significant challenges remain in achieving comprehensive personalization and delivering optimal recommendations. Many RL-based methods fail to reward agents for exploring intermediary sub-paths that could align with user preferences, thereby overlooking paths that might improve recommendation quality. Addressing these limitations is critical for advancing DRL-based RSs and achieving a deeper understanding of user preferences.

The proposed approach introduces the Guided Representation of User Preferences via Path Embedding Propagation (RUPPEP) method, a novel framework designed to enhance RSs using DRL. This study constructs a KG from product attributes, features, and historical purchase data, which serves as the environment for the MDP in the DRL framework. To enable effective navigation of the KG, the method generates embeddings that facilitate the exploration of various path patterns within the KG structure. DRL-powered RSs leverage the rich, structured knowledge within KGs to improve recommendation accuracy [16]. The RUPPEP framework uniquely rewards agents for pursuing intermediate pathways that align with user preferences derived from historical purchase behavior, ensuring that recommendations are contextually relevant and user-specific. Key innovations of the RUPPEP method include a soft reward mechanism to incentivize meaningful exploration, user-conditional action pruning to streamline decision-making, and multi-hop scoring to capture complex relationships across multiple KG nodes. These techniques collectively enable the system to learn and adapt dynamically to evolving user preferences, advancing the state of KG-based RSs and addressing critical challenges in personalization and recommendation quality.

The contributions of this research can be summarised as follows.

● We proposed an RL-based approach utilizing the RUPPEP component, which leverages historical purchasing data and integrates advanced techniques, including a soft reward mechanism, user-conditional action pruning, and multi-hop scoring. These innovations enhance the model's ability to capture nuanced user behavior patterns and interpret complex interactions within the data, enabling the generation of more accurate and contextually relevant recommendations. By dynamically adapting to user preferences, this approach addresses key challenges in personalization and advances the effectiveness of RL-driven RSs.
● We extended the concepts introduced and presented in the prior conference paper [17], offering a comprehensive and detailed exploration of the RUPPEP component. The extended work delves deeper into its underlying methodologies, providing a thorough analysis of its design, implementation, and contribution to enhancing RSs.
● We performed rigorous evaluations across multiple Amazon e-commerce domains to assess its effectiveness. These evaluations demonstrate the framework's capability to infer user preferences accurately and deliver optimal recommendations. The results highlight the robustness and adaptability of the approach in diverse real-world scenarios, underscoring its potential for advancing state-of-the-art RSs.

The paper is structured as follows: Section 2 provides a literature review, including background and related works. Section 3 elaborates on the proposed methodology. Results and discussions are presented in Section 4, and Section 5 concludes the paper with a summary and future directions.

2.0 BACKGROUND AND RELATED WORKS

This section presents an overview of key terminologies associated with KGs and RL, alongside a review of relevant studies that explore the application of DRL in RSs.

2.1 Knowledge Graph

A KG is a directed graph comprising a set of vertices V (nodes) and edges E (relationships). In this structure, nodes represent entities, while edges denote relationships between them, typically expressed as subject-predicate-object triplets. Each edge is represented as <$e_h e_h$, rr, $e_t e_t$>, where $e_h e_h$ is the head entity, rr is the relationship, and e_t e_t is the tail entity [18]. This construction enables the KG to serve as a structured representation of information, capturing intricate relationships between real-world entities.

KGs are often enriched with labeled relationships that provide context or semantic meaning to the connections between entities [19]. Furthermore, KGs frequently incorporate an ontological schema that organizes entities and relationships into hierarchical or categorical structures. This schema adds a layer of organization, enhancing the graph's ability to model complex domains and providing a structured, semantically rich view of the information it contains [20]. These attributes make KGs valuable for diverse applications, including semantic search, RSs, and natural language understanding.

2.2 Knowledge Graph Embedding

Due to the nature of the Heterogeneous Information Network (HIN), the KG may contain enormous amounts of data and face performance issues. While KGs prove proficient in representing structured factual data, they are difficult to manipulate due to the large-scale and complicated graph structure. Consequently, the challenge lies in effectively and efficiently extracting and harnessing valuable information from extensive KGs. This becomes particularly crucial for subsequent tasks such as link prediction [21], entity classification [22], and recommendation generation [23].

To address this formidable challenge, the machine learning (ML) community has introduced the KG embedding (KGE) technique [21]. The core concept behind KGE is to embed entities and relationships within a KG into a lower-dimensional space, typically represented as vectorial embeddings. These embeddings preserve the semantic meaning and relational structure of the original KG. Subsequently, the learned embeddings for entities and relationships are leveraged to tackle downstream tasks, including KG completion [24], question answering [25], and entity classification [22].

Overall, KGE embeds a KG into dd-dimensional space. The word embedding means vectorization of the information into low dd-dimensional vectors. The low-dimensional embedding still preserves the inherent property of the graph and can be quantified by semantic meaning or high-order proximity [26].

2.3 Reinforcement Learning

In RL, an autonomous agent optimizes its performance in a task by interacting dynamically with its environment. As explained in [27], the agent perceives its environment through sensory inputs and influences it via actuators. Instead of relying on explicit expert guidance, RL agents evaluate their actions based on feedback from a reward function. Through iterative interactions, the agent transitions between states, selects actions, and receives rewards determined by the environment in response to those actions. The key goal of the agent is to maximize the cumulative reward, also referred to as the return, over the course of its interactions. This reward-driven learning framework enables RL agents to adapt to complex and dynamic environments, making it a powerful approach for solving sequential decision-making problems.

2.4 Reinforcement Learning Framework

The RL framework comprises Environment State, Agent, Actions, and Rewards. MDPs are the foundational framework for modeling sequential decision-making problems involving an RL agent, where actions influence immediate rewards, the subsequent states, and future rewards. An MDP is defined by the tuple <S, A, T, Rw, Y S, A, T, Rw, Y>, where SS is the set of all possible states of the environment, *A* denotes the set of actions that the agent can consider in each state, TT is the state transition function, describing the probability of transitioning between states given an action, $RwRw$ is the reward function, which defines the immediate reward associated with each state-action pair, and YY is the discount factor that balances the importance of immediate rewards versus future rewards [28]. This structured formulation enables systematic modeling of complex environments and allows RL agents to learn optimal policies by maximizing cumulative discounted rewards over time.

The following describes a few methods for implementing RL.

2.4.1 Q-Learning

Q-learning [29], an influential outcome of early RL research, is an off-policy and Temporal Difference (TD) learning algorithm. It enables the agent to iteratively update a Q-table to derive an optimal policy, using a target policy for this purpose. Meanwhile, a separate behaviour policy, often employing the ε-greedy strategy, facilitates semi-random exploration of the environment. The primary objective of Q-learning is to approximate the optimal action-value function, $q*$, through direct estimation. This function represents the maximum expected cumulative reward achievable by taking a specific action in a given state and following the optimal policy thereafter.

Q-learning employs a dual-policy approach: the target policy guides the optimization of $q*$, while a separate behavior policy, often implemented using a greedy exploration strategy, facilitates the agent's semi-random exploration of the environment. This separation allows the algorithm to balance exploration and exploitation effectively. By directly estimating the action-value function, Q-learning provides a robust mechanism for solving discrete-state, discrete-action RL problems, laying the groundwork for more advanced RL algorithms.
The Q-learning algorithm can be formulated as follows.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha * \left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (1)$$

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha * \left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

where the equation denotes that the state $s_t s_t$ explores the environment with a behaviour policy based on the values in the Q-table at timestep tt, i.e., it performs action aa, the reward RR and a new state $s_{t+1}s_{t+1}$ is obtained based on the feedback from the environment. The equation is executed to update and obtain the latest Q-table. This process continues by updating the new state $s_{t+1}s_{t+1}$ after completing the above operation until the termination time tt.

2.4.2 REINFORCE

The REINFORCE algorithm [30] adopts a unique approach to policy optimization by directly learning a parameterized policy, bypassing the need for value function estimation. Leveraging the Monte Carlo method, REINFORCE updates policy parameters based on complete trajectories and their estimated returns, enabling it to optimize policies using full episode data.

This algorithm uses neural networks to parameterize policies, with the networks taking states as inputs and producing probability distributions over the action space. These probability distributions are then used to guide action selection, allowing the agent to explore the environment efficiently. By concentrating exclusively on policy optimization, REINFORCE provides a foundational framework for policy-gradient methods and has significantly influenced the evolution of more advanced policy-based reinforcement learning techniques.
The policy $\pi\pi$ is parameterized with a set of weights $\theta\theta$ so that $\pi(s;\ \theta) \equiv \pi(s)$,$\pi(s;\ \theta) \equiv \pi(s)$, which is the action probability distribution on the state, and REINFORCE is updated with:

$$\triangle\, \omega_{i,j} = \alpha_{i,j}\left(r - b_{i,j}\right) \frac{\vartheta}{\vartheta\, \omega_{i,j}} \ln g_i \triangle\, \omega_{i,j} = \alpha_{i,j}\left(r - b_{i,j}\right) \frac{\vartheta}{\vartheta\, \omega_{i,j}} \ln g_i \quad (2)$$

where $a_{i,j}a_{i,j}$ is a non-negative learning factor, rr denotes the discounted reward value, and $b_{i,j}b_{i,j}$ is a representation function of the state for reducing the variance of the gradient estimate. Williams [30] points that $b_{i,j}b_{i,j}$ could have a profound effect on the convergence speed of the algorithm. $g_i g_i$ is the probability density function for randomly generating unit activation-based actions.

2.4.3 Actor-Critic

The Actor-Critic algorithm [31] represents a hybrid RL framework that integrates a parameterized policy model (Actor) with a value function estimator (Critic) to enhance both learning efficiency and stability. The Critic evaluates the expected cumulative reward for a given state-action pair, providing feedback to guide the Actor in optimizing its policy via policy gradient methods. This dynamic feedback loop enables the Actor to make more informed and adaptive decisions, enhancing its ability to explore the environment effectively while maximizing long-term rewards. By combining policy-based and value-based methodologies, the Actor-Critic architecture achieves a balance between exploration and exploitation, ensuring robust policy learning. This architecture serves as a foundational framework for numerous advanced RL algorithms, offering flexibility and scalability for solving complex sequential decision-making problems.

2.5 Methods Leveraging KG with RL Techniques

Prominent RL-based approaches, such as PGPR [7], KGDQN [9], EKar [10], ReMR [11], RKGR-RNS [12], and HR-RL-KG [13], address recommendation problems by modeling them within the framework of MDPs. These approaches employ RL to train agents using sophisticated techniques, including path-searching algorithms, state transition strategies, terminal conditions, and reward mechanisms. By leveraging RL, these approaches enable the discovery of interpretable paths that connect users to recommended items. This path-based reasoning enhances the recommendation process's transparency and contributes to the development of explainable and user-centric RSs.

2.6 Discussions and Research Gaps

Xian [7] proposed PGPR, which integrates MDP with RL to navigate multi-hop paths within KGs for recommendations. This method enhances transparency by generating interpretable paths connecting users to items. However, PGPR faces challenges related to computational overhead, limited personalization, and suboptimal product recommendations. Moreover, its reliance on path probability for reasoning path selection overlooks other critical factors, limiting its adaptability. Song [10] introduced EKAR, which combines deep RL and MDP to derive user-preference-based reasoning paths. While this approach advances path interpretability, it requires extensive personalized policy training data. Additionally, its binary reward mechanism (+1, 0, or -1) for terminal entity states lacks intermediate feedback, reducing its effectiveness in suggesting optimal products and their reasoning paths. The prioritization of recommendations is predominantly driven by path probabilities or terminal rewards, restricting their overall impact.

In a subsequent study, Xu [9] developed KGDQN, which integrates KGs into a deep Q-network (DQN) to enhance item recommendation. This approach mitigates some limitations of traditional methods by introducing a non-binary reward strategy. However, it inherits core limitations of DQN, including overestimation bias and instability. The recommendation process primarily depends on maximizing the product of rewards and probabilities, which restricts its ability to provide optimal product suggestions and reasoning paths consistently. Wang [11] proposed ReMR, leveraging ontology-based and instance-level KGs to model multi-level user interests. This approach improves the understanding of complex user preferences, leading to more accurate recommendations. However, its reliance on hierarchical KG structures introduces additional computational challenges. Zhang [12] introduced RKGR-RNS, which focuses on enhancing reward mechanisms in RL for explainable recommendations. The model effectively captures user preferences by incorporating a negative sampling method, enabling more refined recommendations. Despite this, its performance is constrained by the quality of the sampling strategy and its dependency on reward prioritization. Song [13] proposed HR-RL-KG, which employs RL to enable agents to traverse diverse paths within KGs. By leveraging TransD embeddings, this method enhances reasoning and recommendation accuracy. However, its reliance on embedding strategies and path traversal introduces scalability concerns in large and dynamic KG environments.

Several approaches have adopted DRL over KGs to enhance the explainability of recommendations and assess their effectiveness. For instance, MES [14] introduces the concept of a maximum explainability score to evaluate the quality of generated explanations. While multiple explanations may exist for a given recommendation, MES identifies the explanation with the highest explainability score as the most effective. This scoring mechanism not only improves the interpretability of recommendations but also aids in prioritizing recommendations based on their explainability, thereby aligning recommendation strategies with user-centric goals.

Table I delves deeper into a critically assessed summary of key advancements and drawbacks of RL-based research in this area, which uses similar datasets intended to be used in this research and identifies the research gaps that are the focal points of this study.

Table 1: Existing research - key contributions and limitations

Model	Key Contributions	Limitations
EKAR [10]	Harnesses the capabilities of deep RL within an MDP framework to generate interpretable pathways driven by user preferences.	a. Implementing this approach necessitates a large volume of personalized policy training data to attain optimal performance. b. A key limitation is the lack of an intermediary reward mechanism, as it relies on binary rewards (+1, 0, or -1) based on the fulfillment status of paths involving terminal entities. c. As a result, EKAR encounters significant challenges in generating refined product recommendations and corresponding explanations, as its decision-making process primarily depends on path probabilities or rewards to prioritize recommended products.
KGDQN [9]	Incorporates KGs into a DQN framework for item recommendations. The DQN employs a RL algorithm that learns optimal actions through trial and error to solve tasks.	a. This approach inherits limitations from the DQN model, such as overestimation bias and training instability. b. It continues to struggle with identifying the most optimal recommended products and their corresponding reasoning paths.
PGPR [7]	An RL agent was trained for pathfinding, framing the recommendation problem as an MDP to identify a meaningful path linking user-item pairs within the KG. The agent was	a. However, the approach faces several challenges, including high computational complexity due to intricate KG structures.

	trained to sample paths between users and items by meticulously designing the path-search algorithm, transition strategy, terminal conditions, and RL rewards. During the prediction phase, PGPR generates recommended items for users while providing specific paths to interpret the reasoning process.	b. It may also struggle with effectively personalizing recommendations to individual user preferences. c. There is a risk of generating suboptimal recommendations. d. Additionally, it may overlook relevant factors by relying solely on path probabilities in the reasoning process..
ReMR [11]	Utilizes ontologies and instance KGs to comprehensively represent user interests across multiple levels, capturing complex hierarchical relationships. By incorporating multi-level reasoning paths, ReMR enhances the modeling of user preferences, leading to more refined recommendations.	a. Fails to reward agents for following intermediate paths that align with user preferences. b. Lacks a quantitative evaluation of explainability. c. Faces challenges in prioritizing recommended products effectively.
HR-RLKG [13]	The research presents a method that uses RL to enhance agents' ability to navigate diverse paths and utilize the underlying KG, employing the TransD embedding structure.	a. The choice of embedding method has a significant impact on recommendation performance

In summary, each model addresses varied aspects of user preference adaptation. However, challenges persist regarding personalization and optimal product recommendations. This research tackles these challenges by integrating user preferences into traversal paths, enhancing personalization. It optimizes knowledge representation and recommendation by identifying key traversal paths that align with user interests.

3.0 PROPOSED METHOD

This section introduces the proposed RUPPEP method, designed to address challenges in capturing user preferences and delivering optimal product recommendations. By leveraging historical purchase data, the method uncovers valuable traversal patterns and product preferences, enabling more precise recommendations within the KG. The RUPPEP component incorporates an enhanced reward function and an optimized path-finding algorithm to model user preferences during training effectively. Before detailing the methodology, this paper outlines the foundational concepts integral to the proposed approach.

3.1 Edge-Pruning Techniques

Edge pruning plays a critical role in refining the proposed component by addressing uncertainties related to the alignment of specific items with users' preferences. Due to the heterogeneous and complex nature of the KG, certain connections may exhibit higher levels of uncertainty. The edge pruning mechanism mitigates this challenge by evaluating the relevance of potential actions. Specifically, this approach computes matrix multiplication scores by integrating user embeddings with corresponding relationship embeddings and all candidate actions. Actions are then ranked based on their scores in descending order, retaining only the most relevant ones while pruning the rest. By eliminating less pertinent edges, this process effectively reduces the action space within the RL environment, enhancing both computational efficiency and recommendation precision.

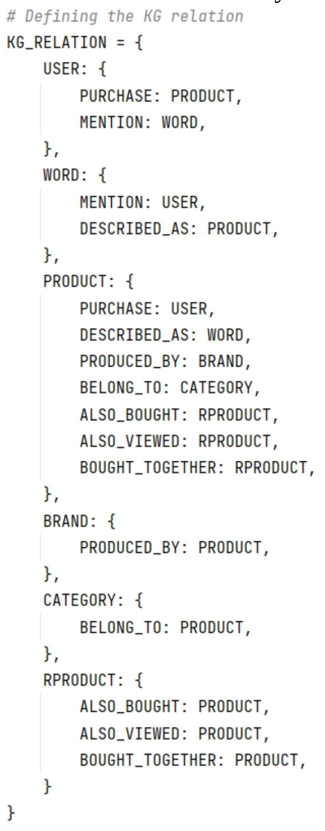
3.2 Ontological Schema

This study employs a JSON-based representation to define the ontological schema of the KG for the respective datasets, as depicted in Fig. 1 (a). In this schema, the outer keys correspond to the SUBJECT nodes, the inner keys represent the PREDICATE relationships, and the associated inner values denote the OBJECT nodes within the Subject-Predicate-Object (SPO) triplet structure of the KG. This structured representation facilitates the systematic organization and retrieval of relational data, enabling efficient knowledge modeling and integration.

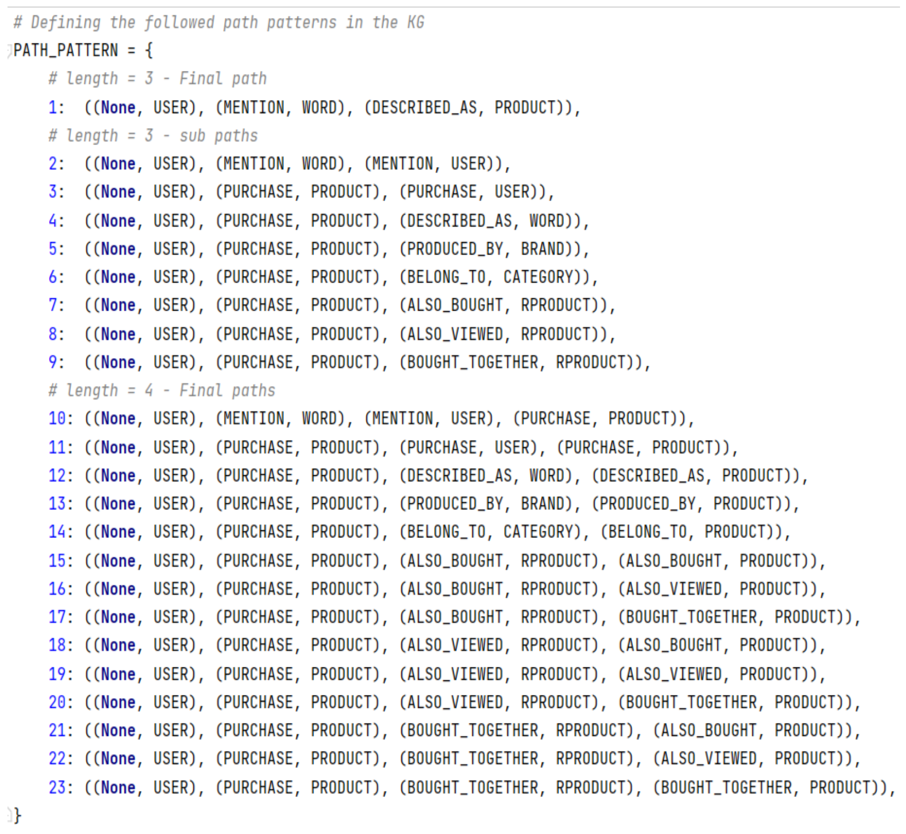
The entity USER is connected to the entity PRODUCT through the PURCHASE relationship, indicating transactional interactions. The USER entity may also be linked to other entities, such as WORD, through relationships like MENTION, which represent descriptive or contextual references associated with products. Users may mention multiple words related to various products and may also have a history of purchasing certain products. Similarly, other relationships among entities exist, as illustrated in Fig. 1 (a), providing a comprehensive representation of interactions within the KG.

3.3 Path Patterns

This study incentivizes the RL agent to discover optimal pathways that lead to favorable outcomes, such as user purchases or interactions, by tracing sequences of interactions within the KG. The research focuses on path patterns, as depicted in Fig. 1 (b), encompassing two- or three-hop transitions, with the USER node serving as the origin and the final hop converging on the PRODUCT node. This approach systematically explores all potential relationships and sub-nodes within the KG, enabling the agent to effectively capture user preferences and enhance recommendation accuracy.



```
# Defining the KG relation
KG_RELATION = {
    USER: {
        PURCHASE: PRODUCT,
        MENTION: WORD,
    },
    WORD: {
        MENTION: USER,
        DESCRIBED_AS: PRODUCT,
    },
    PRODUCT: {
        PURCHASE: USER,
        DESCRIBED_AS: WORD,
        PRODUCED_BY: BRAND,
        BELONG_TO: CATEGORY,
        ALSO_BOUGHT: RPRODUCT,
        ALSO_VIEWED: RPRODUCT,
        BOUGHT_TOGETHER: RPRODUCT,
    },
    BRAND: {
        PRODUCED_BY: PRODUCT,
    },
    CATEGORY: {
        BELONG_TO: PRODUCT,
    },
    RPRODUCT: {
        ALSO_BOUGHT: PRODUCT,
        ALSO_VIEWED: PRODUCT,
        BOUGHT_TOGETHER: PRODUCT,
    }
}
```



```
# Defining the followed path patterns in the KG
PATH_PATTERN = {
    # length = 3 - Final path
    1:  ((None, USER), (MENTION, WORD), (DESCRIBED_AS, PRODUCT)),
    # length = 3 - sub paths
    2:  ((None, USER), (MENTION, WORD), (MENTION, USER)),
    3:  ((None, USER), (PURCHASE, PRODUCT), (PURCHASE, USER)),
    4:  ((None, USER), (PURCHASE, PRODUCT), (DESCRIBED_AS, WORD)),
    5:  ((None, USER), (PURCHASE, PRODUCT), (PRODUCED_BY, BRAND)),
    6:  ((None, USER), (PURCHASE, PRODUCT), (BELONG_TO, CATEGORY)),
    7:  ((None, USER), (PURCHASE, PRODUCT), (ALSO_BOUGHT, RPRODUCT)),
    8:  ((None, USER), (PURCHASE, PRODUCT), (ALSO_VIEWED, RPRODUCT)),
    9:  ((None, USER), (PURCHASE, PRODUCT), (BOUGHT_TOGETHER, RPRODUCT)),
    # length = 4 - Final paths
    10: ((None, USER), (MENTION, WORD), (MENTION, USER), (PURCHASE, PRODUCT)),
    11: ((None, USER), (PURCHASE, PRODUCT), (PURCHASE, USER), (PURCHASE, PRODUCT)),
    12: ((None, USER), (PURCHASE, PRODUCT), (DESCRIBED_AS, WORD), (DESCRIBED_AS, PRODUCT)),
    13: ((None, USER), (PURCHASE, PRODUCT), (PRODUCED_BY, BRAND), (PRODUCED_BY, PRODUCT)),
    14: ((None, USER), (PURCHASE, PRODUCT), (BELONG_TO, CATEGORY), (BELONG_TO, PRODUCT)),
    15: ((None, USER), (PURCHASE, PRODUCT), (ALSO_BOUGHT, RPRODUCT), (ALSO_BOUGHT, PRODUCT)),
    16: ((None, USER), (PURCHASE, PRODUCT), (ALSO_BOUGHT, RPRODUCT), (ALSO_VIEWED, PRODUCT)),
    17: ((None, USER), (PURCHASE, PRODUCT), (ALSO_BOUGHT, RPRODUCT), (BOUGHT_TOGETHER, PRODUCT)),
    18: ((None, USER), (PURCHASE, PRODUCT), (ALSO_VIEWED, RPRODUCT), (ALSO_BOUGHT, PRODUCT)),
    19: ((None, USER), (PURCHASE, PRODUCT), (ALSO_VIEWED, RPRODUCT), (ALSO_VIEWED, PRODUCT)),
    20: ((None, USER), (PURCHASE, PRODUCT), (ALSO_VIEWED, RPRODUCT), (BOUGHT_TOGETHER, PRODUCT)),
    21: ((None, USER), (PURCHASE, PRODUCT), (BOUGHT_TOGETHER, RPRODUCT), (ALSO_BOUGHT, PRODUCT)),
    22: ((None, USER), (PURCHASE, PRODUCT), (BOUGHT_TOGETHER, RPRODUCT), (ALSO_VIEWED, PRODUCT)),
    23: ((None, USER), (PURCHASE, PRODUCT), (BOUGHT_TOGETHER, RPRODUCT), (BOUGHT_TOGETHER, PRODUCT)),
}
```

(a) KG-Relation – an Ontological Schema

(b) Path-patterns for the RL agent

Fig 1(b) illustrates the diverse path patterns that the RL environment promotes for the agent to explore and learn, capturing various interaction patterns associated with the user. The diagram encompasses 23 distinct path combinations, including intermediary sub-paths, which ultimately converge on the PRODUCT node as the final target. These path patterns guide the RL agent in identifying optimal traversal strategies, aligning with user preferences and enhancing the recommendation process.

3.4 Reward Function

The reward function (*Rw*) quantifies the value associated with a traversal path culminating in a recommended product for a specific user, as formalized in Eq. 3. A higher *Rw* value signifies a more optimal traversal path, reflecting better alignment with user preferences. $PATH_{PATTERN}PATH_{PATTERN}$ denotes a set of predefined path structures that guide the RL agent's exploration within the KG. These patterns are designed to steer the agent's actions toward generating recommendations that closely correspond to user preferences, thereby enhancing the quality and relevance of the suggested products.

$$Rw = \begin{cases} 0, & len(path) \leq 2 \\ +ve\ Rewards, & len(path)\ \geq 3\ and\ follows\ PATH_{PATTERN} \\ High + ve\ Rewards, & +ve\ Rewards\ and\ node\ type\ is\ \text{"Product"} \\ -ve\ Rewards, & len(path)\ \geq 3\ and\ NOT\ follows\ PATH_{PATTERN} \end{cases} \tag{3}$$

$$Rw = \begin{cases} 0, \\ +ve\ Rewards, \\ High + ve\ Rewards, \\ -ve\ Rewards, \end{cases}$$

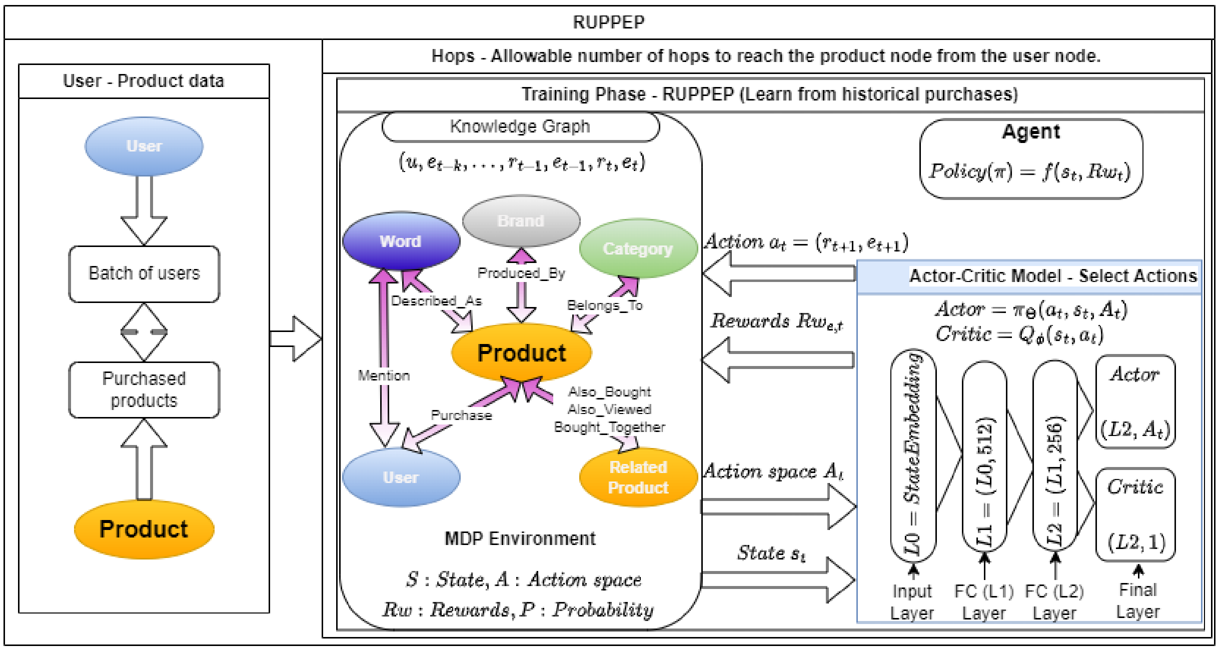


Fig 2: RUPPEP –Representation of User Preferences via the Path Embedding Propagation - A method to learn user preferences from prior purchases

3.5 Path-finding Algorithm: RUPPEP – Actor-Critic Model

The RUPPEP component leverages the Actor-Critic model to learn user preferences by incorporating historical purchasing data and employing advanced strategies such as a soft reward mechanism, user-conditional action pruning, and multi-hop scoring. These strategies collectively aim to capture user-specific path preferences and produce highly relevant recommendations. As detailed in Algorithm 1, the primary objective is to train an RL agent within a KG environment to maximize cumulative rewards by effectively sampling reasoning paths tailored to each user, ultimately converging on previously purchased items. To enhance computational efficiency, graph pruning techniques are employed, utilizing reward values as a heuristic to narrow the search space and expedite the identification of optimal paths.

Following is the elaboration of the steps.

- Develop an MDP environment

The initial step involves constructing an MDP environment that utilizes the KG and its embeddings (KGE). This environment incorporates essential components, including the state *S*, action *A*, reward *Rw*, transition function *T*, and discount factor *Y*.

- Develop an Actor-Critic model

The subsequent step involves designing an Actor-Critic deep RL model with hidden layers consisting of 512 and 256 neurons. This architecture enables the model to effectively capture and represent intricate patterns in user preferences by processing the KG information across multiple layers, thereby facilitating the RL agent's ability to efficiently learn optimal actions for product recommendations.

- Core processing logic

The core processing logic of the RUPPEP framework, as depicted in Fig. 2, initiates with the identification of a batch of users and the associated data regarding their previously purchased products. These user-product interactions serve as the foundational input for the RL agent, enabling it to identify patterns and user preferences that facilitate the refinement of product recommendations. The RL agent leverages these interactions to optimize its decision-making process, ultimately enhancing the precision.

```
Algorithm 1: RUPPEP – Training an RL Agent to Capture User Preferences via Guided Path Patterns

Input:
● KG ($\mathcal{G}_{X\mathcal{R}}\mathcal{G}_{X\mathcal{R}}$): A KG with Users entities $\mathcal{U}\mathcal{U}$, various relationships R, and a set of Items $\mathcal{V}\mathcal{V}$.
● $PATH_{PATTERN}PATH_{PATTERN}$: Defined path patterns guiding agent traversal.
● Episodes: Number of training episodes.
● Discount Factor (γ): Weight for future rewards.
● Reward Function: Mechanism to compute rewards ( $RwRw$) for actions.
● Pruning Threshold: Threshold to filter irrelevant actions.
● Batch Processing size $(b)b$): A batch of user size for the RL agent training

Output: A better trained RL $ActorCriticActorCritic$ model for personalized recommendations.

// a. Develop an MDP environment
● Create an MDP environment using KG and KGE as the core components.
● Initialize the KG state by considering the embedding vector size and the current hop level.
● Design a function to compute MDP rewards for batch processing.
● Implement a function to determine the MDP action space for batch processing.
● Develop a function to generate the current MDP state during batch processing.

// b. Initialize the Actor-Critic RL model
● Initialize Actor and Critic networks with random parameters.
● Set up a replay buffer to store transitions $(s, a, r, s').(s, a, r, s')$.
● Define the action space based on KG edges.
● Configure an exploration strategy (e.g., ϵ-greedy).

// c. Core processing logic of the RUPPEP component
 For each episode $ee$=1 to Episodes:
● Reset the environment to the initial state $s_0s_0$ (USER node).
● Initialize cumulative reward $Rw_e = 0Rw_e = 0$.
   While the episode is not terminated:
      Edge Pruning:
● Calculate action relevance scores by combining user embeddings, relationship embeddings, and candidate actions.
● Retain top actions exceeding the Pruning Threshold, reducing the action space.
      Action Selection:
● Use the Actor network to choose an action $a_ta_t$.
● Introduce exploration using the ϵ-greedy strategy.
      Action Execution:
● Perform the selected action $a_ta_t$, transitioning to the next state $s_{t+1}s_{t+1}$.
● Receive a reward $Rw_tRw_t$ from the environment via the Reward Function.
      Store Transition:
● Save $(s_t, a_t, Rw_t, s_{t+1}s_t, a_t, Rw_t, s_{t+1})$ in the replay buffer for training.
      Update Episode Reward:
● Accumulate reward: $Rw_e \leftarrow Rw_e + Rw_tRw_e \leftarrow Rw_e + Rw_t$.
Model Training:
```

```
● Sample a minibatch ( $\boldsymbol{bb}$) from the replay buffer.
// Identify the batch users and their prior purchased products for developing the Actor-Critic RL model
//Set of users from $\boldsymbol{\mathcal{UU}}$ of a batch size ($bb$)
a.    {u} ←← select ($\boldsymbol{\mathcal{UU}}$, $bb$)
//Set of purchased products for batch users {u} from the train dataset.
b.    $\{v\}$ ←$\{v\}$ ← get_product ($\boldsymbol{\mathcal{VV}}$, {u}, 'Purchase')

● **Critic Update**: Minimize Temporal Difference (TD) error.
● **Actor Update**: Optimize policy.
**Path Pattern Alignment**:
● Encourage the agent to prioritize actions that align with predefined $Path_{Patterns}Path_{Patterns}$
● Penalize deviations from these patterns to reinforce guided learning.
// Train the agent to adapt to user purchase behaviour. The agent will get maximum rewards if the path follows the set pattern and ends to the i'th targeted products $\{v\}_i\{v\}_i$.
a.    $\boldsymbol{for\ i\ in\ range}\ (\boldsymbol{len}(\{v\}):\boldsymbol{for\ i\ in\ range}\ (\boldsymbol{len}(\{v\}):$
    i.Reset the State, Actions, and Rewards for the set of users {u} to set of i'th products $\{v\}_i\{v\}_i$
    i.For hop in range (3):
        1.    Agent selects the actions out of probable actions in the $\boldsymbol{\mathcal{G}_{X\mathcal{R}}\mathcal{G}_{X\mathcal{R}}}$.
        2.    Environment provides the next set of actions after applying the Edge-pruning techniques, new state, path followed, and the rewards based on the current state and actions.
        3.    The agent minimizes the Actor-Critic model loss based on the rewards received from the MDP environment and updates the model parameters accordingly using the Adam optimizer.

**End of Episode**:
● Record cumulative reward $Rw_eRw_e$ for performance evaluation.

// d. Save the checkpoint of the RL model
● Save the optimized Actor-Critic network parameters.
● Utilize the trained policy for efficient path traversal and user-specific recommendations.
```

The RL agent is trained to navigate connectivity path patterns, constrained to a maximum of three hops to maintain manageable complexity. Through an exploration-exploitation process, the agent learns to traverse a network of interconnected path patterns to reach the target purchased products. These paths consist of a sequence of nodes, with no more than three intermediate hops allowed. The agent's objective is to make decisions (actions) that guide it to the target purchased products while maximizing the cumulative rewards.

During training, the agent iteratively selects actions from the available options in each state, aiming to maximize the cumulative reward. Each step evaluates possible actions and chooses the most likely to lead to the target purchased product, yielding the highest reward. As the agent moves through the connectivity path pattern, it receives intermediate sub-path rewards, which can be positive or negative. Positive rewards reinforce actions that align with the predefined path pattern, directing the agent towards the target product, while negative rewards discourage suboptimal paths.

The Actor-Critic model iterates this process across multiple epochs for all users, combining the advantages of both policy-based (Actor) and value-based (Critic) approaches. This enables the agent to refine its decision-making over time by processing a selected batch of users. The model's learning improves as the agent accumulates rewards and adjusts its exploration strategy based on the paths it follows.

This methodology embodies a proactive strategy to capture user preferences by training an RL agent to navigate through structured path patterns in a KG-based environment. By optimizing reward structures and iterating through interactions with the training dataset, the RL agent effectively uncovers latent user preferences, enhancing the recommendation process.

4.0 RESULT AND DISCUSSION

This section outlines the experimental setup, presents a detailed analysis of the results obtained, and offers insights.

4.1 Experimentation

The experiment used two Amazon e-commerce domain-specific datasets: Cell Phones and Beauty [32]. These datasets encompassed six entities: User, Product, Product's describing words, Related Products, Brand, and Category. The relationships among these entities were diverse, comprising eight different types, as depicted in Fig 3. Users purchase products of specific categories and brands, and products are linked with related products, such as 'also viewed' or 'bought together'. Additionally, the datasets encapsulate associations between products and user-mentioned feature words, further enriching the representation of user-product interactions.

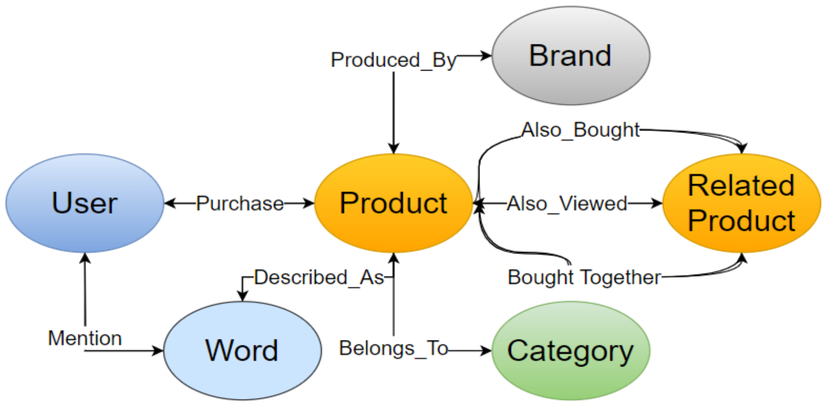


Fig. 3: Knowledge Graph Representation

4.1 Data Descriptions

Table II provides a detailed statistical summary, offering insights into the entities, relationships, and characteristics of the datasets employed in this study. The upper section of the table enumerates the entities present within each dataset, while the lower section examines the relationships between head and tail entities, presenting their mean and standard deviation values. Significantly, the relationships *mention* and *described_as* are prevalent across all datasets, underscoring their critical role in capturing product features. Among the three relationships connecting Products and Related Products, the *also_bought* relationship stands out as the most dominant, highlighting the substantial impact of co-purchasing patterns on user preferences and product recommendations.

Table 2: Data description

Entities	**Description**	**Cell Phones**	**Beauty**
User (U)	Users in RS	27,879	22,363
Product (P)	Candidate Products	10,429	12,101
Feature/Word (W)	Describing words	22,493	22,564
RelatedProduct (RP)	Also bought/viewed	101,287	164,721
Brand (B)	Product's brand	955	2,077
Category (C)	Product's category	206	248
Relations	**Description**	**Mean ± Std Deviation**	
purchase	$U \xrightarrow{purchasepurchase} P$	7 ±± 4.5	8.9 ±± 8.2
mention	$U \xrightarrow{mentionmention} W$	652.1 ±± 1335.8	806.9 ±± 1344.1
described_as	$P \xrightarrow{described\_asdescribed\_as} W$	1,743.2 ±± 3,482.8	1,491.2 ±± 2,554
belong_to	$P \xrightarrow{belong\_tobelong\_to} C$	3.5 ±± 1.1	4.1 ±± 0.7
produced_by	$P \xrightarrow{produced\_byproduced\_by} B$	0.5 ±± 0.5	0.8 ±± 0.4
also_bought	$P \xrightarrow{also\_boughtalso\_bought} RP$	56.5 ±± 35.8	73.6 ±± 30.7
also_viewed	$P \xrightarrow{also\_viewedalso\_viewed} RP$	1.2 ±± 4.3	12.8 ±± 9
bought_together	$P \xrightarrow{bought\_togetherbought\_together} RP$	0.8 ±± 0.8	0.7 ±± 0.7

4.3 Knowledge Graph Development

The first step is to develop the KG. The KG was developed by using entities, as described in Fig 3, as nodes. Our datasets had reviews of products that users had purchased. Before utilizing the review comments in our KG consideration, we ensured that the review comments have a threshold of 0.2 and having word frequency threshold of less than or equal to 5000. The reason for opting for such criteria is to ensure that the KG is meaningful and assist in recommending the right content to the user.

4.4 Data Visualization

The KG, generated after preprocessing the dataset, consists of various nodes and relationships, as illustrated in Fig 4. It provides a detailed view of the relationships that a particular user (12341) has within the Beauty KG. This user has two types of relationships: “PURCHASE” and “MENTIONS.” User 12341 has purchased several products, indicated by the “PURCHASE” relationships, and has mentioned specific product feature words during these purchases, as denoted by the “MENTIONS” relationships. This graphical representation aids in understanding how user interactions with products and features are captured in the KG.
The subsequent nodes may have multiple relationships with various other nodes. Fig 5 illustrates one such node, a product (9227). This product, previously purchased by user 12341, has further relationships of the types: PURCHASE, DESCRIBE_AS, BELONGS_TO, PRODUCED_BY, ALSO_VIEWED, ALSO_BOUGHT, and BOUGHT_TOGETHER.

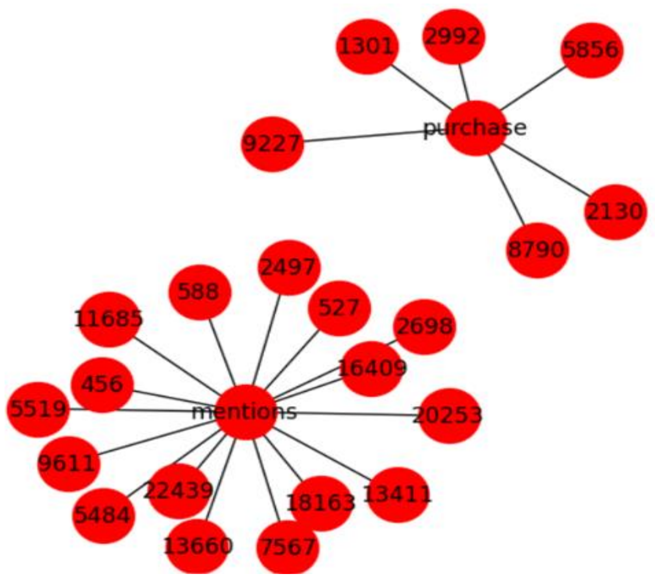


Fig. 4:Beauty KG – A relationship view for user 12341

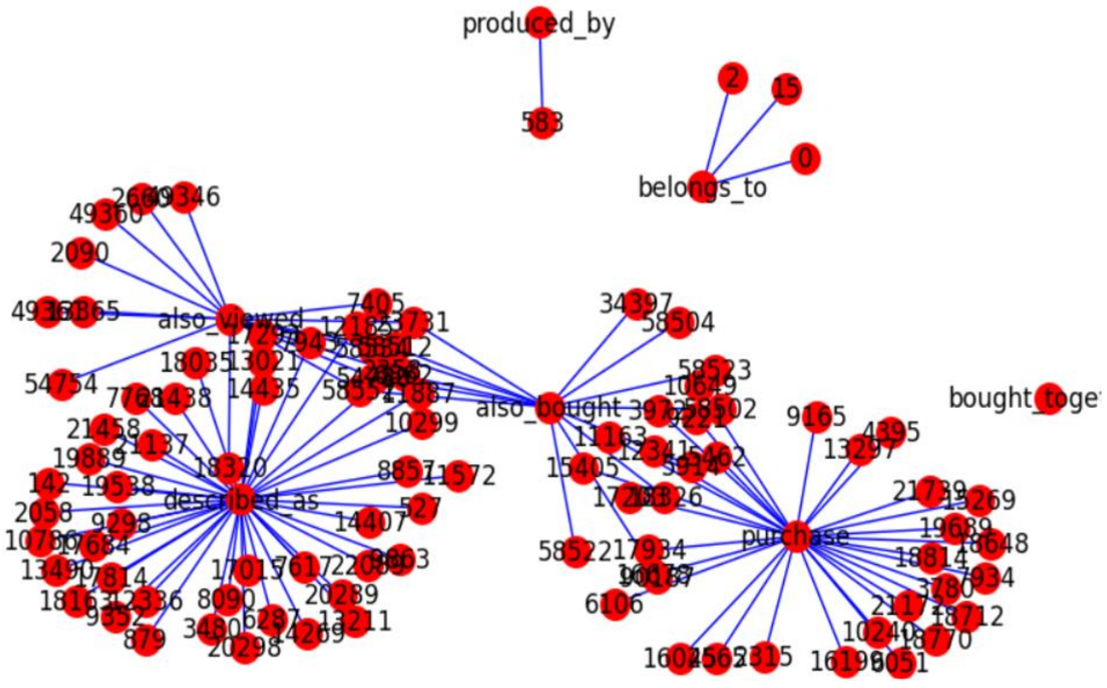


Fig. 5: Beauty KG – A relationship view of Product 9227 bought by user 12341

4.5 Knowledge Graph Embedding Development

Following the construction of the KG, embeddings for the KG were generated using specific hyperparameters to enhance the representation of entities and relationships. The embedding model was trained over 30 epochs with a batch size of 32, and checkpoints were configured to occur every 10,000 steps to ensure incremental improvements in embedding quality. To mitigate potential data loss in case of interruptions, the model was designed to resume seamlessly from the latest checkpoint. Over the course of 30 epochs, the model iteratively refined the embeddings, resulting in enhanced representation fidelity and improved performance in downstream recommendation tasks.

The embedding model continuously updated parameter weights, aiming to minimize the smooth loss function associated with embedding generation. This iterative optimization ensured that the vector representations of entities and relationships effectively captured the structural and semantic properties of the KG. By reducing information loss during the embedding process, the model facilitated efficient processing and accurate predictions.

The primary goal of the embedding model was to generate robust vector representations of the KG, encapsulating its state and relational dynamics. This step was pivotal to the study, as the quality and effectiveness of the generated embeddings had a direct impact on the performance and precision of the recommendation model.

4.6 Actor-Critic RL Model Development

After constructing the KGE, the next phase involved developing the RL model. The RL model was trained using the following hyperparameters: 100 epochs with a batch size of 32, and checkpoints were configured to be saved every 5,000 steps to ensure consistent progress tracking and model improvement. To safeguard against data loss due to interruptions, the model was designed to resume training from the last checkpoint. A learning rate of 0.0001 was employed, with a maximum of 250 allowable actions per state. The model was further constrained to a maximum of three hops to manage complexity effectively. Additionally, the RL architecture incorporated two hidden layers with 512 and 256 neurons, respectively, to facilitate robust learning and representation.

The primary objective of this RL model was to train an agent and its corresponding embeddings to effectively capture user path preferences. The model aimed to develop a policy capable of accurately predicting the next recommendations for users by leveraging their historical purchasing patterns and semantic relationships within the

KG. The training process emphasized aligning the RL agent's actions with user preferences, rewarding actions that followed predefined path patterns, and minimizing the loss associated with the RL model.
The Actor-Critic RL framework played a pivotal role in this study, combining the advantages of policy-based and value-based learning to refine the agent's decision-making process. The effectiveness of this RL model directly influenced the quality and precision of the recommendation outcomes, underscoring its critical role in achieving the study's objectives.

4.7 Results

The RUPPEP algorithm effectively captures user preferences, leading to significant improvements in the accuracy and reliability of the recommender system. For performance evaluation, the proposed method was compared against state-of-the-art approaches, specifically PGPR [7], ReMR [11], and HR-RL-KG [13]. These baseline models leverage RL technologies and are widely recognized for their effectiveness in recommendation tasks.

To establish the superiority of the RUPPEP algorithm, a comparative analysis was conducted using established evaluation metrics, including Normalized Discounted Cumulative Gain (NDCG), Hit Ratio (HR), Recall, and Precision. These metrics, commonly used in previous research, are the most effective way to assess the efficacy of recommendation methods. Table III highlights the experimental results, illustrating the superior performance of the proposed method across multiple datasets. Compared to the best-performing baseline, HR-RL-KG, the RUPPEP algorithm demonstrated notable improvements of 13.26% and 15.43% in NDCG for the Cell Phones and Beauty datasets, respectively.

These performance enhancements extended to other key metrics, including Recall, HR, and Precision, underscoring the robustness and generalizability of the RUPPEP algorithm. The results conclusively establish the proposed method's effectiveness in delivering enhanced recommendation performance across diverse datasets, further affirming its potential as a significant advancement in the domain of explainable RSs.

Table 3: Comparison of effective metrics with baselines

Baseline Models	**Cell Phones**				**Beauty**			
	NDCG	Recall	HR	Prec.	NDCG	Recall	HR	Prec.
PGPR [7]	5.04	8.41	11.9	1.27	5.45	8.32	14.4	1.71
ReMR [11]	5.29	8.72	12.5	1.34	5.87	8.89	15.6	1.91
HR-RL-KG [13]*	5.43	8.93	12.5	1.36	5.83	8.73	14.78	1.94
Ours	**6.15**	**10.39**	**14.5**	**1.6**	**6.73**	**10.46**	**17.34**	**2.18**
% Gain	**13.26**	**16.35**	**16**	**17.65**	**15.43**	**19.81**	**17.32**	**12.37**

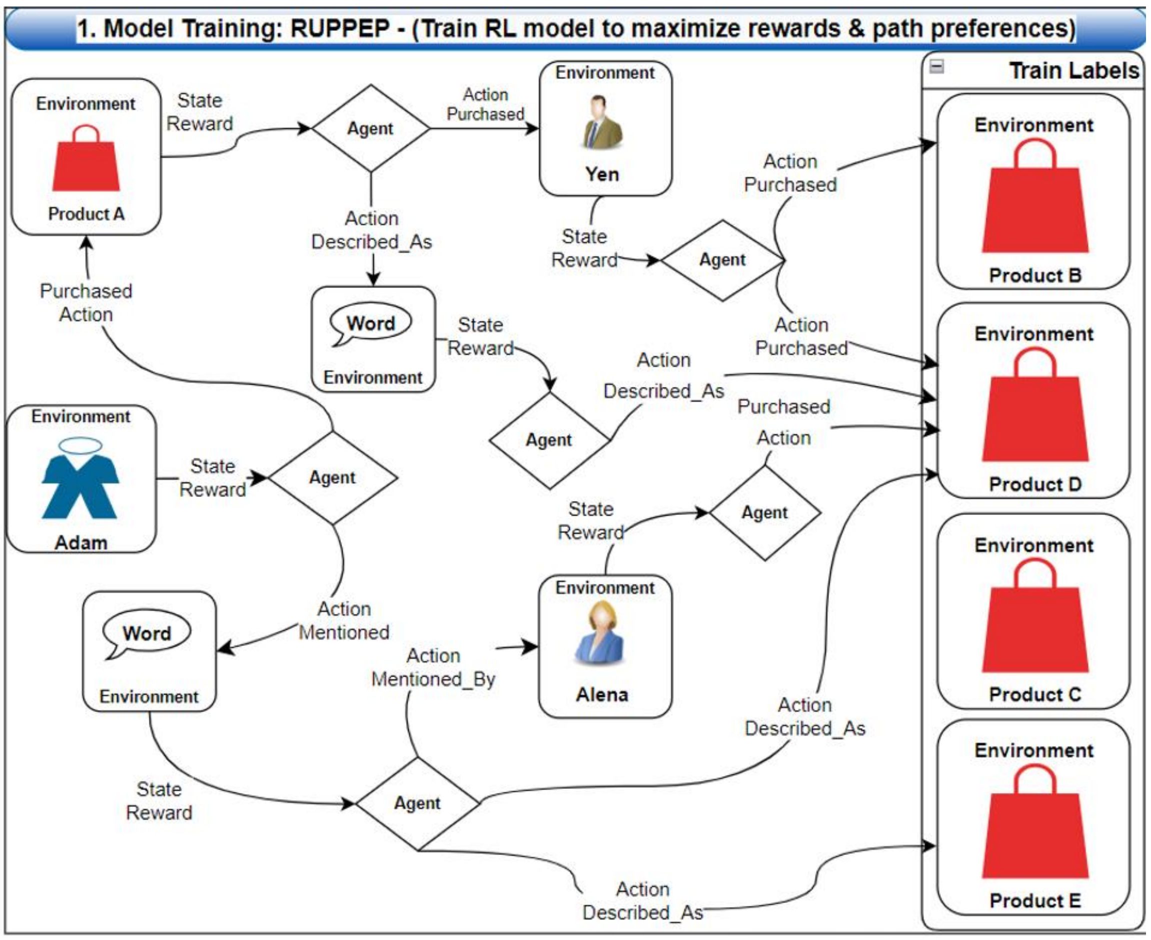


Fig. 6: RUPPEP - An illustration of a case study

4.8 RUPPEP – A User Case Study

The RUPPEP component trains the Actor-Critic model by utilizing historical purchasing information in a batched manner. The goal is to learn path preferences and generate enticing recommendations and tailored explainability. The RUPPEP component operates by leveraging the RL environment to motivate the agent's actions through a reward mechanism. This motivation is rooted in encouraging the agent to follow specific path patterns that lead to products the user has previously purchased. These purchased products, referred to as "Train Labels," are visibly marked in the diagram. Historical purchase information enables the framework to train the model, refining its recommendations by learning from and adapting to user preferences.

Fig 6 illustrates the working mechanism of the RUPPEP component. Consider a user named Adam, who has previously purchased Product B, Product D, and Product E. The KG contains multiple paths that Adam might have followed to reach these products, representing different decision patterns that contribute to understanding his preferences and behaviour.
The diagram shows limited scenarios using relationships such as PURCHASE, DESCRIBE, and MENTION. In the MDP environment, the agent can choose a path leading to a PURCHASE action or a MENTION action for Adam. The PURCHASE action indicates that Adam previously bought "Product A", while the MENTION action suggests a word mentioned by Adam.

Based on probabilities and path-traversal algorithms designed under the RUPPEP framework, the agent selects the most suitable path, leading to decisions influenced by received rewards and new probable actions from the intermediary node. For example, suppose the agent chooses the "Product A" path. In that case, it may need to decide further based on available actions, such as whether another user named Yen also purchased Product A or if certain words describe Product A.

The RUPPEP component recognizes that there might be multiple ways for Adam to reach the same products, and some of these paths might have enticed the RL agent to maximize rewards. Essentially, when Adam chose a particular path that led to a product, the RL agent received positive rewards for successfully guiding Adam. The

RL agent learns from these patterns, internalizing the relationships between products and the routes Adam has historically favoured. By doing so, the agent effectively deciphers the path affinities that Adam typically follows to discover products he might have a predilection for and get an awareness of user preferences.

5.0 CONCLUSION AND FUTURE WORKS

In conclusion, this study addresses the challenges of personalization in RSs by proposing the RUPPEP approach, which integrates KGs with RL. The RUPPEP method effectively utilizes insights derived from historical purchase data to capture user preferences more accurately. The RL model, developed using the RUPPEP algorithm, facilitates the prediction of critical parameters for generating personalized product recommendations. Experimental results demonstrated substantial improvements across key evaluation metrics, including NDCG, HR, Recall, and Precision, highlighting the positive impact of the proposed approach on RS performance.

This work opens several avenues for future exploration. First, subsequent research could further investigate incorporating advanced algorithms and methodologies to enhance recommendation capabilities. Potential directions include integrating sophisticated machine learning models, advanced natural language processing (NLP) techniques, and using explicit user feedback to refine personalized recommendations and generate interpretable explanations. Future work could also focus on reducing the computational complexity and improving the scalability of the solution to tackle larger problems. The current method is computationally intensive. These advancements would continue to improve the effectiveness and user-centricity of RSs.

5.0 ACKNOWLEDGEMENT

Universiti Kebangsaan Malaysia supports this work through the Prime Impact Fund under Grant DIP-2020-017.

REFERENCES

[1] N. A. Osman, S. A. Mohd Noah, M. Darwich and M. Mohd, "Integrating contextual sentiment analysis in collaborative recommender systems," *PLoS ONE,* vol. 16, no. 3, p. e0248695, 2021, https://doi.org/10.1371/journal.pone.0248695.

[2] S. M. Al-Ghuribi, S. A. Mohd Noah, M. A. Mohammed, N. Tiwary and N. I. Y. Saat, "A Comparative Study of Sentiment-Aware Collaborative Filtering Algorithms for Arabic Recommendation Systems," in *IEEE Access*, vol. 12, pp. 174441-174454, 2024, https://doi.org/10.1109/ACCESS.2024.3489658.

[3] N. Liu and J. Zhao, "Recommendation System Based on Deep Sentiment Analysis and Matrix Factorization," *IEEE Access,* vol. 11, pp. 16994-17001, 2023, https://doi.org/10.1109/ACCESS.2023.3246060.

[4] M. R. A. KHALIL and A. A. BAKAR, "A Comparative Study of Deep Learning Algorithms in Univariate and Multivariate Forecasting of the Malaysian Stock Market (Kajian Perbandingan Algoritma Pembelajaran Mendalam dalam Peramalan Univariat dan Multivariat Pasaran Saham Malaysia)," *Sains Malaysiana,* vol. 52, no. 3, pp. 993-1009, 2023, http://doi.org/10.17576/jsm-2023-5203-22.

[5] Y. Zhang and X. Chen, "Explainable Recommendation: A Survey and New Perspectives," *Foundations and Trends® in Information Retrieval,* vol. 14, no. 1, pp. 1-101, 2020, http://dx.doi.org/10.1561/1500000066.

[6] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine,* vol. 34, no. 6, pp. 26-38, 2017, https://doi.org/10.1109/MSP.2017.2743240.

[7] Y. Xian, Z. Fu, S. Muthukrishnan, G. de Melo and Y. Zhang, "Reinforcement Knowledge Graph Reasoning for Explainable Recommendation," in *SIGIR'19: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, New York, NY, USA, Association for Computing Machinery, 2019, pp. 285–294, https://doi.org/10.1145/3331184.3331203.

[8] K. Zhao, X. Wang, Y. Zhang, L. Zhao, Z. Liu, C. Xing and X. Xie, "Leveraging Demonstrations for Reinforcement Recommendation Reasoning over Knowledge Graphs," in *SIGIR '20: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, Association for Computing Machinery, 2020, pp. 239–248, https://doi.org/10.1145/3397271.3401171.

[9] W. Xu, X. Gao, Y. Sheng and G. Chen, "Recommendation system with reasoning path based on dqn and knowledge graph," in *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, Seoul, Korea (South), IEEE, 2021, pp. 1-8, https://doi.org/ 10.1109/IMCOM51814.2021.9377414.

[10] W. Song, Z. Duan, Z. Yang, H. Zhu, M. Zhang and J. Tang, "Ekar: An Explainable Method for Knowledge Aware Recommendation," *arXiv preprint arXiv:1906.09506*, vol. 13, p. 1906.09506, 2022, https://doi.org/10.48550/arXiv.1906.09506.

[11] X. Wang, K. Liu, D. Wang, L. Wu, Y. Fu and X. Xie, "Multi-level Recommendation Reasoning over Knowledge Graphs with Reinforcement Learning," in *WWW '22: Proceedings of the ACM Web Conference 2022*, Virtual Event, Lyon, France, Association for Computing Machinery, 2022, pp. 2098–2108, https://doi.org/10.1145/3485447.3512083.

[12] S. Zhang, Y. Ouyang, Z. Liu, W. Rong and Z. Xiong, "Reinforcement Learning-Based Explainable Recommendation over Knowledge Graphs with Negative Sampling," in *2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta)*, Haikou, China, IEEE, 2022, pp. 1948-1953, doi: 10.1109/SmartWorld-UIC-ATC-ScalCom-DigitalTwin-PriComp-Metaverse56740.2022.00282.

[13] W. Song, T. Wang and Z. Zhang, "Recommendations Based on Reinforcement Learning and Knowledge Graph," in *IEA/AIE 2023: Advances and Trends in Artificial Intelligence. Theory and Applications*, vol. 13925, Cham, Springer, 2023, pp. 313--324, https://doi.org/10.1007/978-3-031-36819-6_28.

[14] N. Tiwary, S. A. Mohd Noah, F. Fauzi and T. S. Yee, "Max Explainability Score–A quantitative metric for explainability evaluation in knowledge graph-based recommendations," *Computers and Electrical Engineering*, vol. 116, p. 109190, 2024, https://doi.org/10.1016/j.compeleceng.2024.109190.

[15] X. Chen, L. Yao, J. McAuley, G. Zhou and X. Wang, "Deep reinforcement learning in recommender systems: A survey and new perspectives," *Knowledge-Based Systems*, vol. 264, p. 110335, 2023, https://doi.org/10.1016/j.knosys.2023.110335.

[16] R. M. Nawi, S. A. M. Noah and L. Q. Zakaria, "Integration of Linked Open Data in Collaborative Group Recommender Systems," *IEEE Access*, vol. 9, pp. 150753-150767, 2021, https://doi.org/ 10.1109/ACCESS.2021.3124939.

[17] N. Tiwary, S. A. Mohd Noah, T. S. Yee and S. Al-Ghuribi, “Advancing Recommender Systems with Deep Reinforcement Learning,” *in the 16th International Conference On Knowledge and Systems Engineering (KSE 2024) & The Sixth International Conference On Information Rrtrieval and Knowledge Management (CAMP 2024),* Kuala Lumpur, 2024.

[18] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong and Q. He, “A survey on knowledge graph-based recommender systems,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 34, no. 8, pp. 3549-3568, 2020, https://doi.org/10.1109/IAEAC50856.2021.9390863.

[19] A. Hogan, E. Blomqvist, M. Cochez, C. d’Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier and others, "Knowledge Graphs," *ACM Computing Surveys (Csur),* vol. 54, no. 4, pp. 1-37, 2021, https://doi.org/10.1145/3447772.

[20] M. Yahya, J. G. Breslin and M. I. Ali, “Semantic Web and Knowledge Graphs for Industry 4.0,” *Applied Sciences,* vol. 11, no. 11, p. 5110, 2021, https://doi.org/10.3390/app11115110.

[21] Z. Sun, Z.-H. Deng, J.-Y. Nie and J. Tang, “RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space,” *arXiv preprint arXiv:1902.10197,* 2019, https://arxiv.org/abs/1902.10197.

[22] D. Yu, C. Zhu, Y. Yang and M. Zeng, “JAKET: Joint Pre-training of Knowledge Graph and Language Understanding,” in *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*, Virtual, 2022, https://doi.org/10.1609/aaai.v36i10.21417.

[23] Y. Xian, Z. Fu, G. De Melo and Y. Zhang, “Reinforcement Knowledge Graph Reasoning for Explainable Recommendation,” in *In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR’19)*, Paris, France, 2019, https://doi.org/10.1145/3331184.3331203.

[24] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston and O. Yakhnenko, “Translating Embeddings for Modeling Multi-relational Data,” in *Advances in Neural Information Processing Systems,* vol. 26, C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Weinberger, Eds., Curran Associates, Inc., 2013, https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

[25] W. Zheng, L. Yin, X. Chen, Z. Ma, S. Liu and B. Yang, “Knowledge base graph embedding module design for Visual question answering model,” *Pattern Recognition,* vol. 120, p. 108153, 2021, https://doi.org/10.1016/j.patcog.2021.108153.

[26] H. Cai, V. W. Zheng and K. C.-C. Chang, “A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications,” *IEEE transactions on knowledge and data engineering,* vol. 30, no. 9, pp. 1616-1637, 2018, 10.1109/TKDE.2018.2807452.

[27] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani and P. P{\'e}rez, "Deep Reinforcement Learning for Autonomous Driving: A Survey," *IEEE Transactions on Intelligent*

Transportation Systems, vol. 23, pp. 4909--4926, 2021, doi: http://dx.doi.org/10.1109/TITS.2021.3054625.

[28] N. Tiwary, S. A. Mohd Noah, F. Fauzi and T. S. Yee, “A Review of Explainable Recommender Systems Utilizing Knowledge Graphs and Reinforcement Learning,” *IEEE Access,* vol. 12, pp. 91999-92019, 2024, 10.1109/ACCESS.2024.3422416.

[29] C. J. Watkins and P. Dayan, “Q-learning,” *Machine Learning,* vol. 8, no. 3, pp. 279-292, 1992, https://doi.org/10.1007/BF00992698.

[30] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning,* vol. 8, no. 3, pp. 229-256, 1992, https://doi.org/10.1007/BF00992696.

[31] V. Konda and J. Tsitsiklis, “Actor-Critic Algorithms,” *Advances in Neural Information Processing Systems,* vol. 12, 1999, https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[32] S. AL-Ghuribi, S. A. Mohd Noah and M. Mohammed, “An experimental study on the performance of collaborative filtering based on user reviews for large-scale datasets,” *PeerJ Computer Science,* p. 9:e1525, 2023, https://doi.org/10.7717/peerj-cs.1525.