FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Exploring Representation Learning in Reinforcement Learning Recommender Systems: Leveraging Knowledge Graphs for Enhanced Contextual Understanding

Guilherme Soares

Mestrado em Engenharia e Ciência de Dados

Supervisor: Zafeiris Kokkinogenis, PhD

Co-Supervisor: Tiago Daniel Sá Cunha, PhD

Co-Supervisor: Carlos Manuel Milheiro de Oliveira Pinto Soares, PhD

October 22, 2025

Exploring Representation Learning in Reinforcement Learning Recommender Systems: Leveraging Knowledge Graphs for Enhanced Contextual Understanding

Guilherme Soares

Mestrado em Engenharia e Ciência de Dados

Aprovado em provas públicas pelo Júri:

Presidente: Prof. Ana Aguiar

Arguente: PhD. João Nuno Vinagre Marques da Silva

Vogal: PhD. Zafeiris Kokkinogenis

October 22, 2025

Abstract

Recommender systems have become an integral part of modern digital platforms, enabling personalized suggestions that increase user engagement. Conventional methods, including collaborative filtering and content-based techniques, depend on user-item interaction data to identify patterns in user behavior. However, these methods face substantial challenges, including data sparsity and cold start problems, as rating matrices are typically highly sparse, with users interacting with only a small fraction of the available items.

In order to address these limitations, this thesis explores the integration of Knowledge Graph embeddings into state representations for offline Reinforcement Learning–based sequential recommender systems. This approach enhances traditional state representations by incorporating TransE embeddings semantic information alongside user interaction history, allowing the system to leverage both collaborative filtering signals and content-based semantic relationships.

Through experimental evaluation, we compare Behavioral Cloning, Batch-Constrained Q-learning, and Conservative Q-learning against the GRU4Rec baseline using session-aware metrics for next-item prediction tasks. The results demonstrate that knowledge graph–enhanced state representations significantly outperform the baseline approaches. Behavior Cloning and Batch-Constrained Q-Learning achieved superior performance compared to GRU4Rec and exhibited faster convergence than purely statistical state representations. Conservative Q-learning showed mixed results, suggesting that its effectiveness varies across different offline reinforcement learning algorithms.

These findings contribute to our understanding of how Knowledge Graphs can enhance state representations in offline deep reinforcement learning frameworks for sequential recommendation and provide insight into the trade-offs between semantic enrichment and computational complexity.

Keywords: Recommender Systems, Knowledge Graphs, Deep Reinforcement Learning

Resumo

Os sistemas de recomendação tornaram-se parte central das plataformas digitais modernas, permitindo sugestões personalizadas que aumentam o engajamento do usuário com a plataforma. Os métodos convencionais, incluindo *collaborative filtering* e *content-based*, dependem de dados de interação entre o usuário e item para identificar padrões em seu comportamento. No entanto, esses métodos enfrentam desafios substanciais, incluindo problemas de escassez de dados e *cold-start*, uma vez que os usuárioes tendem a interagir apenas com uma pequena fração dos itens disponíveis.

Para resolver essas limitações, esta tese explora a integração de grafos de conhecimento em representações de estado para sistemas de recomendação sequenciais baseados em aprendizagem por reforço offline. Essa abordagem aprimora as representações de estado tradicionais, incorporando informações semânticas provenientes do algoritmo TransE juntamente com o histórico de interação do usuário, permitindo que o sistema aproveite tanto os sinais de colaborativos quanto as relações semânticas dos items.

Através de uma avaliação experimental, comparamos os algoritmos Behavior Cloning, o Batch-Constrained Q-learning e o Conservative Q-learning com a base de referência sendo o algoritmo GRU4Rec, utilizando métricas por sessão do usuário para tarefas de previsão do próximo item. Os resultados demonstram que as representações de estado aprimoradas pelo gráfico de conhecimento superam significativamente a referência. O Behavior Cloning e o Batch-Constrained Q-learning alcançaram desempenho superior em comparação ao GRU4Rec e exibiram convergência mais rápida do que representações de estado puramente estatísticas. O Conservative Q-learning mostrou resultados mistos, sugerindo que a eficácia dessa integração varia entre diferentes algoritmos de aprendizagem por reforço offline.

Essas descobertas contribuem para nossa compreensão de como os Gráficos de Conhecimento podem aprimorar as representações de estado em estruturas de aprendizagem por reforço profundo offline para recomendação sequencial e fornecem insights sobre as trocas entre enriquecimento semântico e complexidade computacional.

Keywords: Sistemas de Recomendação, Grafos de Conhecimento, Aprendizado por Reforço Profundo

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Prof. Zafeiris, Dr. Tiago Cunha and Prof. Carlos, for their continuous guidance during this study. dialogues serve as a constant source of inspiration to pursue more knowledge and cultivate my love for this field.

I would also like to thank my parents and brother for all the support given throughout the years. Without you I would not be where I am today.

I would also like to express my profound gratitude to my friends, Julia Otani and Bruno Zabot, for their company in the early start of my stay at FEUP. I could not have reached this stage without you.

Lastly, I thank those who have been there for me for a long time and are now spread around the world for their support, company even when distant, love and guidance.

Guilherme Soares

Contents

List of Figures

List of Tables

Abbreviations

BC	Behavioral Cloning
BCQ	Batch Constrained Q-learning
BFS	Breadth-First Search
CB	Content-Based Filtering
CF	Collaborative Filtering
CQL	Conservative Q-Learning
DCG	Discounted Cumulative Gain
DFS	Depth-First Search
DQN	Deep Q-Network
GRU	Gated Recurrent Unit
IDCG	Ideal Discounted Cumulative Gain
ID	Identifier
KG	Knowledge Graph
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MDP	Markov Decision Process
MRR	Mean Reciprocal Rank
NDCG	Normalized Discounted Cumulative Gain
RL	Reinforcement Learning
RS	Recommender Systems
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
VGAE	Variational Graph Auto-Encoders

Chapter 1

Introduction

1.1 Context and Motivation

Recommender Systems (RS) constitute a fundamental part of modern digital platforms, driving user engagement by providing personalized suggestions in real time. Traditional approaches such as collaborative filtering Su and Khoshgoftaar (2009) and content-based methods Poonam B. Thorat (2015) have been the core part of these systems. Collaborative filtering relies on historical user-item interactions to identify patterns among similar users or items. Content-based methods leverage the attributes of items and user preferences to make recommendations based on feature similarity.

While collaborative filtering has demonstrated great efficacy in scenarios characterized by substantial user interaction data, it frequently encountered the cold-start problem, wherein data is scarce for new users or items. In contrast, content-based methods address this issue by focusing on item features but can lead to a lack of diversity in recommendations Pazzani (1999). Hybrid systems attempt to merge the strengths of both methods Burke (2002) but in some cases struggle to adapt dynamically to changes in user preference over time.

In recent years, the integration of deep learning-based systems into recommendation systems has lead to significant improvement of the quality of recommendations Zhang et al. (2019a). Such systems are able to model complex user-item interactions, often represented in multiple modalities, such as text and images. A key component of these deep learning approaches is the use of embedding techniques Liu et al. (2024), which learn low-dimensional vector representations that capture the latent features and semantic properties of users, items, and their various attributes. Although this integration has been shown to enhance recommendation accuracy and context-awareness, it increased the need for a large amount of high quality data while also reducing the interpretability of each recommendation.

Among the deep learning approaches, graph-based methods explicitly model the inherent graph structure in recommendation data Wu et al. (2022). These methods transform the user-item rating data into a bipartite graph, where users and items are represented as nodes, and their interactions are shown as edges. This graph representation can show complex relationships and

multi-hop connections that are not included in matrix-based methods. Graph Neural Networks (GNNs) Scarselli et al. (2009) have emerged as highly effective methods for recommending tasks Gao et al. (2021), especially in scenarios where relational information and structural patterns are crucial for capturing user preferences and item relationships.

However, GNNs depend on high-quality data in appropriate graph format and typically operate on static graph structures. This static nature limits their ability to adapt to evolving user preferences and conflicting interests over time Wu et al. (2020), potentially leading to suboptimal recommendations in dynamic contexts where user behavior changes continuously.

Recently, the field has seen significant progress in addressing these limitations through the application of Deep Reinforcement Learning (DRL) Chen et al. (2021b), Afsar et al. (2022). DRL's unique approach involves treating recommendation tasks as Markov Decision Processes (MDPs), where user states, system actions (i.e. recommendations), and rewards (i.e. user feedback) are modeled to learn sequential policies that capture temporal dynamics of user preferences. This framework optimizes both immediate user satisfaction and long-term engagement, presenting a promising alternative to address the shortcomings of traditional methods, particularly in handling concept drift and cold start problems.

The integration of Knowledge Graphs (KG) with Reinforcement Learning further enhances this approach. While user-item matrices in collaborative filtering can be seen as simple graphs linking users to items, KG provide rich contextual relationships between entities (users, items, attributes, contexts) with labeled edges that specify the type of connection, such as "user prefers genre", "item belongs to category", or "director worked on movie". This semantic structure enables a Reinforcement Learning agent to reason within the knowledge base Lin et al. (2024), providing more accurate and personalized recommendations.

1.2 Problem definition

Traditional recommendation systems primarily rely on static user-item interaction patterns and struggle to capture the temporal dynamics of evolving user preferences and the semantic relationships that influence decision-making processes Pan et al. (2025). Sequential recommendation systems address the temporal aspect by modeling interaction histories and temporal patterns. However, they often fail to leverage the rich semantic properties present in item attributes, categories, and relationships Wang et al. (2019a). These limitations are particularly evident in scenarios with sparse data or complex behavioral patterns. Standard user-item interaction graphs capture direct sequential relationships but lack the semantic depth provided by knowledge graphs, which encode multi-relational information about entities and their interconnections Guo et al. (2020).

Integrating knowledge graph embeddings into the state representation of reinforcement learning–based sequential recommenders is a promising approach to addressing these limitations. However, the effectiveness of different integration strategies remains unclear, and their impact on ranking performance in sequential recommendation tasks has not been systematically evaluated. Current approaches include concatenating knowledge graph (KG) embeddings with user/item repre-

sentations Zhang et al. (2016), using attention mechanisms to fuse KG information Wang et al. (2019b), employing graph neural networks to propagate embeddings through the KG structure Wang et al. (2018), and leveraging meta-path-based methods to extract relational features Hu et al. (2018). These approaches vary significantly in complexity and design; however, there is limited understanding of which strategies genuinely improve performance versus those that introduce unnecessary computational overhead.

1.3 Research Questions

This thesis examines the impact of integrating knowledge graph embeddings into state representations and ranking performance in reinforcement learning–based sequential recommender systems. A case study utilizing the MovieLens dataset (movielens) will be conducted to examine how semantic information encoded in these embeddings can improve the agent's understanding of user preferences and item relationships for next-item prediction.

Specifically, the study will implement and compare multiple KG integration strategies, mean aggregation, recency-weighted aggregation, and quality-weighted aggregation, across three offline reinforcement learning algorithms: Behavioral Cloning (BC) Pomerleau (1988), Batch Constrained Q-Learning (BCQ) Fujimoto et al. (2019), and Conservative Q-Learning (CQL) Kumar et al. (2020) . The performance of these strategies will be evaluated against GRU4Rec Hidasi et al. (2016a), an established recurrent neural network baseline for sequential recommendation, as well as against statistical feature representations without KG enhancement. Conventional ranking metrics (NDCG@K, MRR, and Recall@K) are used to evaluate both algorithmic advances and the value of integrating semantic information.

The research perspective of this thesis aims to answer the following questions:

- RQ1: In sequential recommendation systems, how do different strategies for integrating knowledge graph embeddings into state representations affect ranking performance?
- RQ2: What is the relative effectiveness of using raw statistical features versus embeddings for enhancing state representations?

1.4 Document Structure

The document is organized as follows:

Chapter 2 - Literature Review provides the foundational knowledge required to understand this research, covering the core concepts of Reinforcement Learning, Recommender Systems, and Knowledge Graphs, along with their key techniques and applications. It ends by presenting related work on the intersection of these three areas, their current limitations and the key trends in the research community.

Chapter 3 - Methodology outlines the experimental framework for integrating knowledge graph embeddings into RL-based sequential recommenders. It details the problem formulation, the architecture of the developed system and how KGs are integrated in the recommendation process

Chapter 4 - Experimental Setup presents the implementation details, dataset selection and processing, KG embeddings training process, offline RL algorithms configuration and evaluation protocol.

Chapter 5 - Results and Analysis covers the experimental results, discusses the implications of different integration strategies, and discuss the conditions under which KG embeddings improve sequential recommendation performance.

Chapter 6 - Conclusions and Future Work summarizes the key contributions and limitations of the current work on KG-enhanced sequential recommendation systems, as well as directions for future research.

Chapter 2

Literature Review

2.1 Recommender Systems

Recommender Systems (RS) represent essential components of contemporary digital platforms, meticulously engineered to optimize user experience by providing personalized recommendations. These systems facilitate navigation of vast information spaces by analyzing user preferences and behavior to recommend pertinent items, including products, films, or social media content. The three primary techniques used in RS are collaborative filtering, content-based filtering, and hybrid approaches, each offering unique strengths for recommendations.

2.1.1 Traditional Approaches

Collaborative Filtering Collaborative filtering (CF) Goldberg et al. (1992) uses the collective preferences of users to generate recommendations. It operates under the assumption that users who have exhibited similar preferences in the past are likely to do so again in the future. For example, in a social media context, if two users consistently approve of similar posts, the system may recommend a new post liked by one user to the other. This is made possible by recording the history of interactions in a user-item rating matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$, where m is the number of users and n the number of items, and each entry R_{ui} denotes the rating or interaction of user u with item i. However, this explicit feedback matrix is highly sparse, as only a fraction of the total set of items is rated by users. The CF's objective is to predict the missing information and complete this matrix using various approaches, such as classical statistical estimators, machine learning or deep learning methods.

Matrix factorization techniques Koren et al. (2009) represent one of the most foundational and influential approaches in collaborative filtering, establishing the mathematical framework that underlies many modern recommendation methods. These techniques learn latent representations $\mathbf{u}_u$ and $\mathbf{v}_i$ for each user u and item i, respectively. This maps the user-item matrix R into a lower-dimensional latent space. The predicted rating is then computed as follows:

$$\hat{R}_{ui} = \mathbf{u}_u^T \mathbf{v}_i$$

Table 2.1 illustrates the structure of a user-item rating matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$, where missing values in the matrix are denoted by "-". The sparsity problem in such matrices is evident in real-world scenarios.

	Item 1	Item 2	Item 3	Item 4	Item 5	$\cdots$	Item n
User 1	5	-	3	-	4	$\cdots$	3
User 2	-	2	-	5	-	$\cdots$	-
User 3	4	4	-	-	3	$\cdots$	2
User 4	-	-	5	2	-	$\cdots$	-
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
User m	1	-	4	-	5	$\cdots$	4

Table 2.1: Example of a sparse user-item rating matrix.

Matrix factorization can be approached through three different learning paradigms: point-wise learning, where the system computes a score for each item individually Liu (2009); list-wise learning, where an optimal ranking order is estimated Shi et al. (2010); and pair-wise learning, where pairs of items are compared to approximate the optimal ranking order Rendle et al. (2012).

However, collaborative filtering faces challenges such as the cold start problem, where limited data on new users or items reduces recommendation accuracy. Additionally, highly sparse user-item matrices can greatly impact performance, requiring advanced algorithms to overcome these limitations.

Content-Based Filtering Content-based filtering (CB) Mooney and Roy (2000) takes a different approach to address some of these collaborative filtering limitations, particularly the cold start problem. It analyzes the attributes of the items and their alignment with user preferences. This technique creates user profiles based on the features of items the users have interacted with, forming a representation of their preferences in a vectorized feature space.

The process typically involves three main steps: item representation, where items are characterized by their attributes (e.g., genre, sections, keywords); user profile construction, where a user's preferences are modeled as a combination of item features from their interaction history; and recommendation generation, where similarity between the user profile and candidate items is computed to rank recommendations Poonam B. Thorat (2015). Classical methods typically represent items using manually engineered feature vectors with explicit item attributes, while modern methods use learning systems to acquire dense vector from implicit information Barkan and Koenigstein (2016). The similarity between items is measured using metrics such as cosine similarity or Euclidean distance.

Figure 2.1 illustrates this complete workflow, showing how user interactions and item features are transformed into vectors for similarity computation. The similarity score is then used to generate recommendations.

To illustrate, if a user interacts with sustainability-related content, the system will analyze the textual features, tags, and categories of the user's historical content to build a profile emphasizing

[H]

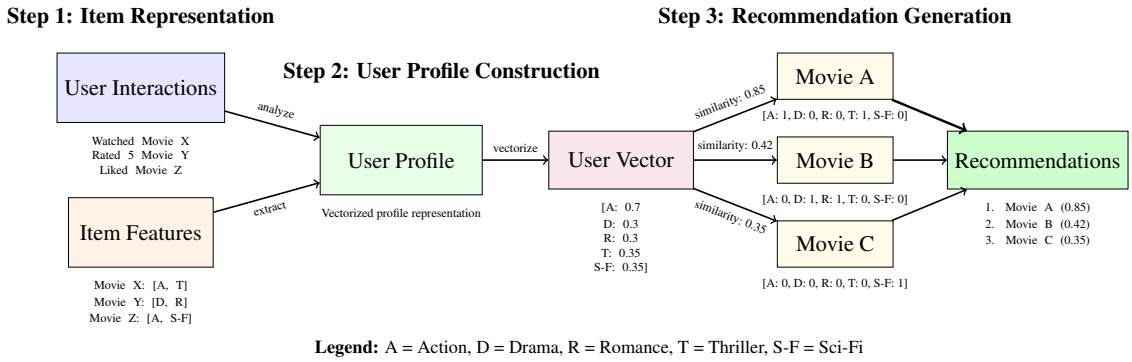

Figure 2.1: Example of a content-based filtering workflow.

environmental themes. The system then suggests new items with similar feature profiles. Although this approach closely aligns recommendations with individual interests and performs well for new items, it often suffers from over-specialization. This reduces diversity, potentially limiting user exposure to novel content and negatively impacting user experience.

Hybrid Systems To address the complementary limitations of these approaches, collaborative filtering's cold start problem and data sparsity issues, and content-based filtering's overspecialization and limited diversity, hybrid systems have emerged. These systems combine multiple strategies to achieve higher accuracy and robustness of recommendations. They can leverage the strengths of each approach: CF's to discover patterns through user similarity and CB's capability to recommend items based on explicit features Poonam B. Thorat (2015).

Several hybridization strategies have been developed to effectively combine these techniques. A common approach is to use switching hybrid systems which select the most appropriate algorithm based on the current context. For example, they might use content-based filtering for new items and collaborative filtering for established items with sufficient interaction data. Another is a weighted system that linearly combine scores from different algorithms, allowing adjustments based on the confidence of the system or data availability Burke (2002). More sophisticated approaches include cascade hybrid systems Pazzani (1999), where one algorithm refines the output of another, and feature combination methods, where features from different approaches are merged into a single machine learning model.

Hybrid approaches that blend collaborative filtering and content-based filtering have proven effective in contexts like social media platforms Sami et al. (2024), where user engagement benefits from both collective preferences and personalized attributes. The 2009's Netflix Prize competition demonstrated the effectiveness of ensemble methods that combine multiple recommendation algorithms, overcoming individual approaches.

Deep Learning-Based Approaches While traditional hybrid systems combine discrete algorithmic approaches, the evolution toward deep learning has enabled more sophisticated integration of multiple recommendation strategies within neural architectures. Deep learning-based RS have

emerged as powerful alternatives to traditional methods, using neural networks to model complex relationships between users, items, and their interactions Zhang et al. (2019a).

These systems are innovative by the way they use embeddings, compact vector representations that turn complex entities like users and items into points in a continuous, low-dimensional space. In these spaces, the relationships between entities are usually preserved through geometric proximity Mikolov et al. (2013b); He et al. (2017). Formally, an embedding function $f : \mathscr{E} \to \mathbb{R}^d$ maps an entity $e \in \mathscr{E}$ to a d-dimensional dense vector, where entities with similar characteristics or behaviors are positioned closer in the embedding space. For example, users with similar viewing preferences will have embeddings $\mathbf{u}_i, \mathbf{u}_k \in \mathbb{R}^d$ that are close in the latent space, while movies of the same genre will have nearby item embeddings $\mathbf{v}_j, \mathbf{v}_l \in \mathbb{R}^d$, enabling recommendation through proximity-based similarity measures such as cosine similarity Sarwar et al. (2001) or learned interaction functions He et al. (2017). Figure 2.2 illustrates an example of a deep learning-based recommendation workflow, where users and items have distinct encoder networks.

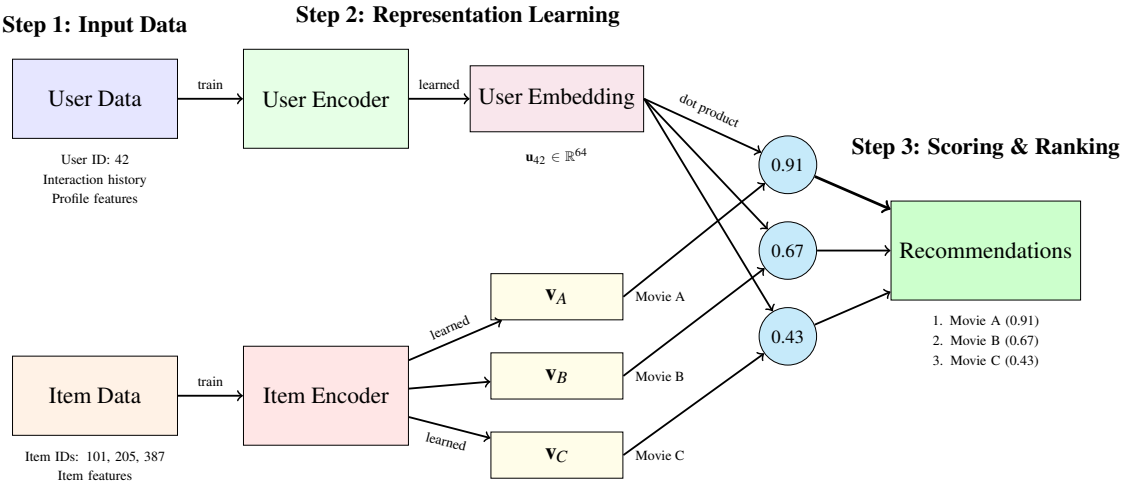

Figure 2.2: Example of a deep learning-based filtering workflow.

By using embeddings, deep learning models can incorporate heterogeneous information sources, including temporal dynamics Hidasi et al. (2015), item metadata Covington et al. (2016), and contextual features Cheng et al. (2016). Modern architectures such as two-tower models Covington et al. (2016), neural collaborative filtering He et al. (2017), and sequence-based approaches using RNNs Hidasi et al. (2015) and transformers Kang and McAuley (2018); Sun et al. (2019) have become industry standards. These approaches have demonstrated great performance in diverse applications within RS, including social media platforms Ying et al. (2018), e-commerce Zhou et al. (2018), and video streaming services Covington et al. (2016), by enabling scalable and highly personalized recommendations.

2.1.1.1 GRU4Rec

A particularly influential sequential recommendation model is GRU4Rec Hidasi et al. (2016b), which pioneered the use of Gated Recurrent Units Cho et al. (2014b) for session-based recom-

mendation. GRU4Rec learns to sequence item interactions within a user's session and derive predictions based on the context of the current session.

Architecture and Item Representation: GRU4Rec represents each item using 1-of-N encoding, where the length of the input vector corresponds to the number of items. Specifically, only the coordinate corresponding to the active item is set to one, while all others are set to zero. The paper found that this direct encoding outperformed embedding layers in their experiments. The core architecture consists of a single GRU layer with additional feedforward layers between the GRU and output, as multiple GRU layers have been shown to consistently degrade performance. The general architecture of GRU4Rec during an event is shown in Figure 2.3.

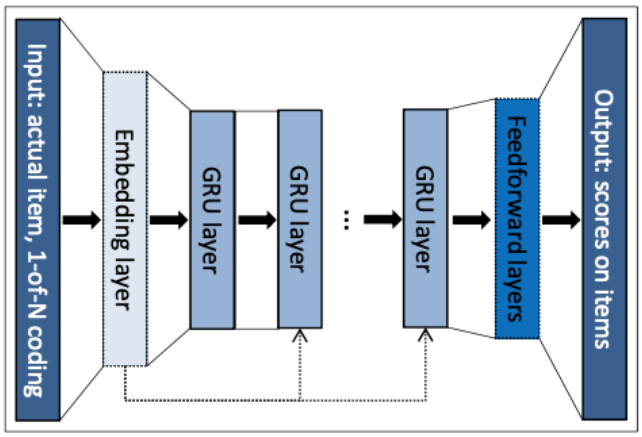

Figure 2.3: GRU4Rec architecture Hidasi et al. (2016b).

Output Sampling and Recommendation Generation: To efficiently manage large item catalogs, GRU4Rec employs a sampling strategy, leveraging items from other training examples within the mini-batch as negative samples. This approach utilizes a popularity-based sampling method, whereby the authors demonstrate that the utilization of mini-batch items as negatives results in a reduction of computational complexity and a natural weighting of sampling by popularity.

Ranking Loss Functions: GRU4Rec introduces two ranking-based loss functions specifically designed for recommendation:

- **BPR (Bayesian Personalized Ranking):** $\mathscr{L}_s = -\frac{1}{N_S}\sum_{j=1}^{N_S}\log(\sigma(\hat{r}_{s,i}-\hat{r}_{s,j}))$, where $\hat{r}_{s,i}$ is the score for the target item and $\hat{r}_{s,j}$ are scores for negative samples. BPR enforces that the ranking of the target item surpasses that of negative items. This is achieved by maximizing the probability that the positive item's score exceeds that of each negative item;

- **TOP1:** A novel loss function: $\mathscr{L}_s = \frac{1}{N_S}\sum_{j=1}^{N_S}\sigma(\hat{r}_{s,j}-\hat{r}_{s,i})+\sigma(\hat{r}_{s,j}^2)$, which approximates the relative rank of the relevant item with sigmoid functions and integrates regularization to ensure that negative scores are maintained near zero;

This approach exhibited substantial enhancements over conventional baselines, such as item-to-item collaborative filtering, attaining 20-30% gains in accuracy metrics. The success of GRU4Rec

in capturing sequential patterns within sessions while addressing practical scalability concerns established it as a foundational method for neural sequential recommendation systems.

2.1.2 Evaluation Frameworks and Metrics

Correct evaluation is essential for determining if a RS can support a platform's strategic objectives and provide real value. Evaluating these systems is complex because they must simultaneously optimize for accuracy, user satisfaction, business metrics, and long-term engagement. Unlike traditional machine learning tasks, which have clear ground truth labels, RS operate in environments where feedback is often implicit, dynamic, and heavily dependent on context Zangerle and Bauer (2022); Herlocker et al. (2004).

Evaluation Methodologies RS can be evaluated in both offline and online settings, with different purposes in each environment for both development and deployment lifecycles Zangerle and Bauer (2022). Offline evaluation uses historical data to assess algorithm performance in controlled conditions, enabling prototyping and comparison between different approaches establishing baseline metrics. In another sense, online evaluation encompasses A/B testing and live experiments, capturing user behavior to measure user-centric metrics Zangerle and Bauer (2022).

Accuracy Metrics Accuracy metrics are fundamental to evaluating RS, measuring how well their predictions align with user's actual preferences. These metrics fall into two main categories: rating prediction accuracy and classification accuracy.

Rating prediction accuracy metrics, such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), evaluate the system's ability to predict exact numerical ratings. While these are valuable for explicit feedback scenarios, they are less relevant for sequential recommendation tasks that typically rely on implicit feedback Ludewig and Jannach (2018).

Classification accuracy metrics treat recommendations as binary classification problems by determining whether items are relevant or not to users. This approach is particularly suitable for implicit feedback systems, where user interactions (e.g., clicks, purchases) indicate positive preferences.

Recall measures the fraction of relevant items that were successfully recommended and is calculated as:

$$\text{Recall@K} = \frac{|R_u^K \cap I_u|}{|I_u|}$$

where R_u^K is the set of top-K recommended items for user u and I_u is the set of items with which the user actually interacted. Recall very important metric in RS, as it quantifies the system's capacity to identify items that users find engaging. In sequential recommendation contexts, it is a metric that evaluates whether the next item a user actually interacted with appear among the top-K predictions. A higher recall value indicates that the system successfully identifies a greater proportion of items that users find relevant, which directly correlates with user satisfaction and engagement Zangerle and Bauer (2022).

Ranking Metrics Ranking metrics evaluate the quality of the order in which recommendations are presented, which is crucial for systems that provide users with ranked lists Shani and Gunawardana (2011). Unlike accuracy metrics, which treat all recommendations equally, ranking metrics assume that users typically focus on the top-ranked items, and that the position of relevant items impact directly the user experience and the effectiveness of the system Joachims (2002), Craswell et al. (2008).

Normalized Discounted Cumulative Gain (NDCG) is the most widely adopted ranking metric Järvelin and Kekäläinen (2002). It evaluates the relevance of ranked recommendations while accounting for position bias. NDCG is formally defined as:

$$\text{NDCG@k} = \frac{\text{DCG@k}}{\text{IDCG@k}}$$

where DCG (Discounted Cumulative Gain) measures the actual gain:

$$\text{DCG@k} = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

and IDCG (Ideal Discounted Cumulative Gain) represents the maximum possible gain achievable with perfect ranking. The term rel_i symbolizes the relevance score of the item at position i, and the logarithmic discounting function $\frac{1}{\log_2(i+1)}$ ensures that relevant items appearing at lower ranks contribute proportionally less to the overall score.

Mean Reciprocal Rank (MRR) Voorhees (1999), focuses specifically on the position of the first relevant item in the ranking. It is formally defined as:

$$\text{MRR} = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{1}{\text{rank}_q}$$

where $|Q|$ is the number of queries (users) and rank_q is the position of the first relevant item for query q. If no relevant item is found, the reciprocal rank is 0. MRR is particularly valuable when users seek to quickly identify at least one relevant item, making it especially suitable for evaluating sequential recommendation systems where the immediate next-item prediction is critical.

Evaluation Challenges The evaluation of RS faces several methodological challenges that can significantly impact the validity of the results. One critical concern is data leakage, which occurs when information from the test set influences the training process, leading to optimistic performance estimation. This issue is exacerbated when ignoring global timeline considerations in offline evaluation settings Ji et al. (2020), as models may learn from user-item interactions that would not be available at prediction time in real-world deployment scenarios.

This is more critical as we consider the evolution of user preferences over time, as items that suffered from the cold start problem during training start to get really popular over the time Adomavicius and Tuzhilin (2005). Without proper temporal splitting, models can learn from these future popularity signals in evaluation datasets with extended time periods, creating an unrealistic

advantage that would not exist in actual deployment scenarios Campos et al. (2014); Schnabel et al. (2016). This violates the fundamental assumption of collaborative filtering: that item characteristics remain relatively stable Koren (2010).

Additionally, research shows that higher offline metrics do not necessarily lead to improvements in online performance Krauth et al. (2020). Although offline metrics are generally correlated with online performance across different environments, this relationship can vary significantly. Consequently, small improvements in predictive performance may result in insignificant enhancements to the actual user experience.

The choice between implicit (e.g. time spent on item page) and explicit (e.g. ratings) feedback evaluation adds another layer of complexity because implicit feedback requires a different interpretation and metric adaptation than explicit ratings do. Explicit feedback is more accurate but less abundant than implicit feedback, as users may spend time on one item, but not consume it. Besides, explicit feedback encompasses both positive and negative feedback, while implicit feedback is generally interpreted as positive only.

These challenges have created a demand for more sophisticated approaches to understanding and modeling user-item interactions. Traditional collaborative filtering methods rely primarily on explicit ratings and struggle to capture the complex relationships between users, items, and their evolving preferences over time. These limitations have led to a evolving adoption of representation learning techniques in RS, which are now fundamental to encoding rich, multidimensional relationships.

2.2 Representation Learning

Representation learning has emerged as a foundational paradigm in modern machine learning. It enables systems to automatically discover meaningful patterns in raw data instead of relying on manually engineered features. This section explores the core principles of representation learning and examines techniques relevant to recommender systems, particularly knowledge graph embeddings.

2.2.1 Fundamentals of Representation Learning

The goal of representation learning is to automatically generate superior features by discovering the underlying patterns and structures in raw data Bengio et al. (2013). This is realized through the acquisition of low-dimensional vector representations that encapsulate the fundamental attributes of the original space, which is often complex and high-dimensional. The core principle of representation learning is that similar entities should have similar representations Mikolov et al. (2013b). This enables models to generalize beyond explicitly observed patterns, infer relationships, and make predictions about unseen data combinations LeCun et al. (2015).

Traditional machine learning approaches require domain expertise for feature engineering, which involves manually crafting representations from raw data Domingos (2012). For example, early RS often used features such as user demographics and item categories Adomavicius and

Tuzhilin (2005), Pazzani and Billsus (2007). However, this manual process is very demanding and usually domain-specific. It often fails to capture the complex, nonlinear relationships inherent in high-dimensional data Bengio et al. (2013).

Representation learning automates the feature discovery process by learning transformations that map raw input data into vectorized representations Bengio et al. (2013), LeCun et al. (2015). This process is done with a mapping $f : X \to Z$ that transforms input data $x \in X$ from a high-dimensional, sparse, or discrete space into a dense, continuous, lower-dimensional space $z \in Z \subseteq \mathbb{R}^d$, where d is typically much smaller than the initial dimension size. This learned representation should capture the semantic relationships of the data, while discarding noise and redundancy.

The learning process usually is done by optimizing an objective function that forces the learned representation to preserve the properties of the initial data Bengio et al. (2013). These objective functions can be categorized into several types:

Reconstructive objectives, such as those used in autoencoders, aim to minimize the reconstruction error between the original input and its decoded representation. The objective function typically takes the form $\mathscr{L}_{rec} = \|x - \text{decode}(\text{encode}(x))\|^2$, where x is the input data vector or feature representation, and the encoder-decoder architecture learns to compress and reconstruct x from a lower-dimensional latent space Hinton and Salakhutdinov (2006); Vincent et al. (2008).

Predictive objectives learn representations by optimizing the model's ability to predict target variables or future observations. In RS, this often involves predicting user-item interactions, such as $\mathscr{L}_{pred} = \sum_{(u,i)} \ell(y_{ui}, f(u,i))$, where ℓ is a loss function, y_{ui} is the true interaction, and $f(u,i)$ is the predicted interaction based on learned representations He et al. (2017); Rendle et al. (2009).

Contrastive objectives enforce that similar entities have closer representations while dissimilar ones are pushed apart in the embedding space. These typically follow the form $\mathscr{L}_{contrast} = -\log \frac{\exp(\text{sim}(z_i,z_j)/\tau)}{\sum_k \exp(\text{sim}(z_i,z_k)/\tau)}$, where $\text{sim}(\cdot,\cdot)$ is a similarity function, τ is a temperature parameter, and the objective maximizes similarity between positive pairs while minimizing it for negative pairs Chen et al. (2020); He et al. (2020).

In the context of RS, representation learning addresses several fundamental challenges, such as the sparsity found in Collaborative Filtering methods, reduced impact of the cold start problem in items with few historical data, and the scalability in very large systems Zhang et al. (2019b). A significant benefit of deep learning-based RS in comparison to classical CF approaches is their capacity to internally automate the learning of these representations during the training process. This capability eliminates the need for manual feature engineering, enabling the capture of complex, nonlinear user-item interactions He et al. (2017), Cheng et al. (2016). In modern systems, the network of users and items is often approached as a graph problem, demanding graph-based representations that can capture multi-hop relationships and structural properties Wu et al. (2022), Wang et al. (2019c).

2.2.2 Fundamentals of Graph Representation Learning

An emerging approach in the area is to model RS as graphs, wherein users and items are represented as nodes, and their interactions, such as purchases, ratings, or views, form the edges of a

bipartite graph Wu et al. (2022). In this formulation, user nodes exclusively connect to item nodes and vice versa, thereby capturing implicit or explicit feedback through edge definitions. This structural framework enables the transformation of the recommendation task into a link prediction problem: predicting potential future user-item interactions based on the existing connectivity patterns in the graph Liben-Nowell and Kleinberg (2003); van den Berg et al. (2017).

This graph-based perspective offers both conceptual and practical benefits. It enables the modeling of high-order relationships by traversing multiple hops in the graph, thereby capturing indirect interactions that are frequently overlooked in conventional matrix factorization approaches. For instance, a user who has not directly interacted with an item may still have significant connections to it via similar users or co-interacted items.

In the RS community, graph learning is most commonly approached as a link prediction problem Hamilton (2020), where the goal is to predict missing or future edges between users and items in a bipartite graph. Given a set of users and items with existing interaction patterns, the task is to infer potential new connections that represent likely user preferences or item relevance. Link prediction problems are frequently addressed through the utilization of proximity-based methodologies, which capitalize on graph structural characteristics, or embedding-based approaches that facilitate the learning of latent representations of nodes Martínez et al. (2016); Zhang and Chen (2018).

Proximity-based Approaches Proximity-based methods learn node representations by leveraging structural properties and topological features of the graph Hamilton (2020). These methods generate representations based on graph statistics and connectivity patterns rather than optimizing embeddings directly.

Structural similarity measures form the foundation of proximity-based representation learning. The Jaccard coefficient measures node similarity as

$$s_{xy}^{Jaccard} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

where $\Gamma(x)$ represents the neighborhood of node x Lü and Zhou (2011). The common neighbors metric $CN(x,y) = |\Gamma(x) \cap \Gamma(y)|$ provides a simpler way to measure structural proximity. These measures directly translate to feature vectors that represent nodes based on their local graph structure.

Degree-based features incorporate node importance into representations. The preferential attachment score $PA(u,v) = |\mathcal{N}(u)| \times |\mathcal{N}(v)|$ captures the likelihood of connection between high-degree nodes Barabási and Albert (1999). In RS, this means using representations that show how popular items are and how active users are. This helps the model understand which popular items get more interactions.

Path-based representations leverage multi-hop connectivity patterns. The Adamic-Adar index $AA(u,v) = \sum_{w \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{\log |\mathcal{N}(w)|}$ weights common neighbors by their rarity, giving more weight to the connections through less popular intermediary nodes Adamic and Adar (2003).

In the context of bipartite graphs in RS, these methods are used to generate node representations by projecting the bipartite structure onto user-user or item-item graphs. Users are represented by their structural similarity to other users through shared items, while items are represented by their co-occurrence patterns. This facilitates the acquisition of collaborative signals using graph topology Zhou et al. (2007).

Embeddings-based Approaches Modern graph embedding techniques are based on the adaptation of natural language processing methods, particularly the Word2Vec model Mikolov et al. (2013a)). Word2Vec showed that meaningful word representations could be learned by predicting co-occurring words within sliding windows of text. The research community then transferred this perspective to graph domains by treating random walks as sentences, where nodes correspond to words and each step provides the context for learning node representations. Two approaches are known as bases in the area: DeepWalk Perozzi et al. (2014) and Node2Vec Grover and Leskovec (2016).

DeepWalk was the first to adapt the word2vec's approach to graphs. It works by simulating random walks starting from each node, collecting its neighboring nodes as its context and applies the Skip-gram model to learn embeddings. The goal is to maximize the probability of seeing nearby nodes for any given node so that structurally or functionally similar nodes end up with similar embeddings.

Node2Vec is built on top of DeepWalk, adding flexibility to the way the algorithm explores the graph. Rather than walking randomly, Node2Vec introduces two parameters that allow the walks to behave more similarly to a breadth-first search (BFS) or a depth-first search (DFS), depending on the assigned values. This allows Node2Vec to capture local similarity or structural similarity, depending on which is more meaningful for the specific application.

While random walk-based methods learn embeddings through neighborhood sampling, an alternative paradigm has emerged through graph autoencoders, which learn representations by reconstructing the graph structure though learned embeddings. Traditional graph autoencoders, such as VGAE (Variational Graph Auto-Encoders) Kipf and Welling (2016), focus on node-level embeddings, encoding each node and reconstruct the adjacency matrix through pairwise dot products.

However, there has been a shift in the latest research moving towards graph-level autoencoders, such as GRALe Krzakala et al. (2025). Basing themselves in Evoformer's Jumper et al. (2021), they leverage attention-based architectures to capture both local patterns and global graph properties in a attempt to learn a unified representations for entire graphs. This marks a significant shift from node-centric to graph-centric representation learning. This shift enables applications such as graph classification, graph generation, and cross-graph comparison in various application fields.

Despite these advances, graph representation learning remains challenging. The unstructured nature of graph data makes applying standard machine learning techniques quite difficult. The large variety of possible graph structures creates scalability issues. The need to preserve multiple

types of structural information simultaneously often leads to low representation quality. Furthermore, graphs in real-world applications tend to be dynamic, heterogeneous, and noisy, which turns the learning process even more difficult.

2.2.3 Knowledge Graphs

Traditional graphs represent relationships between entities through simple edges, lacking explicit semantic information. However, KGs augment this representation by incorporating structured knowledge about the real world through rich semantic relationships Hogan et al. (2021).

Formal Definition A Knowledge Graph is formally defined as a directed, labeled, multi-relational graph $G = (E, R, T)$, where:

- E is the set of entities (nodes)
- R is the set of relation types (edge labels);
- $T \subseteq E \times R \times E$ is the set of triples (h, r, t) representing facts, where h is the head entity, r is the relation, and t is the tail entity;

A key distinction between KGs and traditional bipartite user-item graphs used in collaborative filtering lies in their inherent heterogeneity and multi-relational nature. This characteristic enables KGs to represent a wide spectrum of entity types and relationship semantics Wang et al. (2017). Each triple (h, r, t) encodes a factual statement, such as ("The Matrix", "directed_by", "Lana Wachowski") or ("Action", "is_genre_of", "The Matrix"), as illustrated in Figure 2.4.

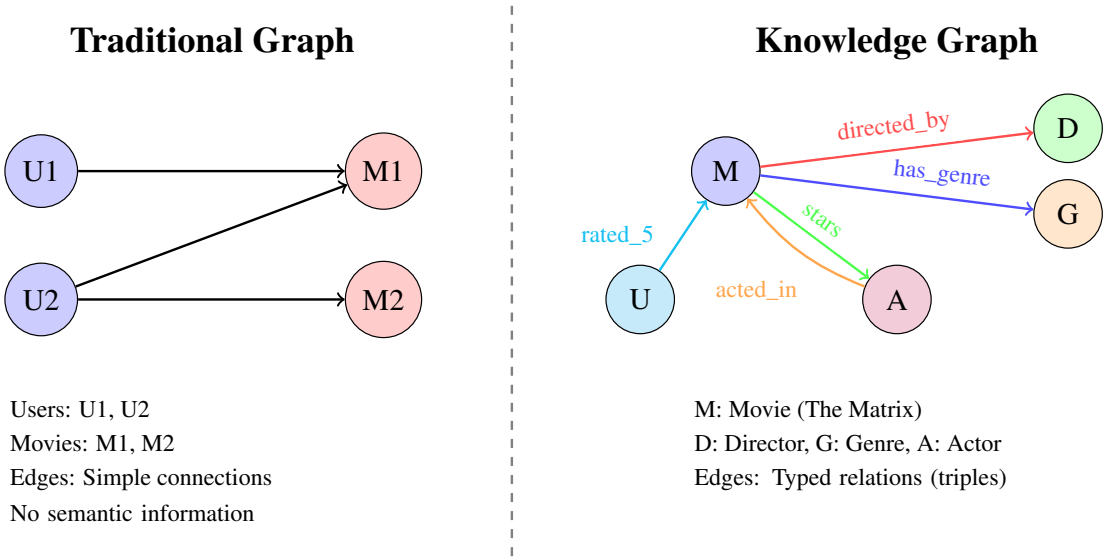

Figure 2.4: Comparison between traditional graphs and knowledge graphs.

Semantic Richness Knowledge Graphs include contextual metadata including entity types, attributes, temporal information, and hierarchical relationships Paulheim (2017). This enables representation of complex domain knowledge such as:

- **Taxonomic relationships**: $(e_1, \text{is\_a}, e_2)$ representing categorical hierarchies;
- **Functional relationships**: $(e_1, \text{works\_for}, e_2)$ representing roles and affiliations;
- **Temporal relationships**: $(e_1, \text{founded\_in}, 1995)$ representing time-dependent facts;
- **Compositional relationships**: $(e_1, \text{part\_of}, e_2)$ representing structural dependencies;

They are constructed by extracting information from several different sources, from unstructured sources, such as articles Wu and Palmer (2018), videos Krishna et al. (2017), sensors Barnaghi et al. (2012), to structured databases, such as SQL tables Sequeda and Miranker (2013). Common techniques used in the construction process include entity recognition Lample et al. (2016), relationship extraction Zeng et al. (2014) and KG completion Nickel et al. (2016). Known RS databases often treated as Knowledge Graphs are MovieLens Harper and Konstan (2015) and Last.fm Schedl (2016).

At disposal, knowledge bases, such as DBpedia Auer et al. (2007) and Wikidata Vrandečić and Krötzsch (2014), are relevant tools for the advancement of studies involving Knowledge Graphs. Inside the RS area, KGs play an important role of including external entities into the systems, enabling the capture of complex semantic relationships. Lately, Large Language Models have been used as a tool to complete missing links in Knowledge Graphs Yao et al. (2024) or initially generate them Chen and Bertozzi (2023).

Knowledge Graphs are composed by several different relational patterns, which include:

- **Symmetric (Antisymmetric)**: A relation is *symmetric* if it is in both directions, that is, if the head entity is related to the tail, then the tail is equally related to the head. In contrast, a relation is *antisymmetric* if both directions can only hold when the entities are the same. For example, *"is similar to"* is symmetric, while *"is older than"* is antisymmetric.

$$\text{Symmetric:} \quad r(h,t) = r(t,h) \qquad \text{Antisymmetric:} \quad r(h,t) \wedge r(t,h) \Rightarrow h = t$$

- **Inverse**: Two relations r_1 and r_2 are inverse if one implies the other in reverse order. For instance, if *"user follows (author)"*, then the inverse is *"author is followed by (user)"*.

$$r_1(h,t) \Leftrightarrow r_2(t,h)$$

- **Composition (Transitive)**: Relations can be composed across multiple steps. For example, if *"item belongs to category"* and *"category belongs to department"*, then the composed relation is *"item belongs to department"*.

$$r_1(h,m) \wedge r_2(m,t) \Rightarrow r_3(h,t)$$

- **1-to-N, N-to-1**: In 1-to-N, one head entity relates to multiple tails (e.g., *"author writes book"*), while N-to-1 means multiple heads relate to one tail (e.g., *"book has genre"*).

$$\text{1-to-N:} \quad r(h,t_1), r(h,t_2), \ldots, r(h,t_n)$$

$$\text{N-to-1:} \quad r(h_1,t), r(h_2,t), \ldots, r(h_n,t)$$

Knowledge Graph Embeddings To take advantage of the relational structure of Knowledge Graphs in machine learning tasks, a common strategy is to transform the graph into a continuous vector space through embeddings. These embeddings aim to preserve the semantics of entities and their relationships, enabling their use in downstream models such as recommendation, link prediction, and reasoning. Several embedding algorithms have been proposed to model the relations shown in Section 2.2.3, such as:

- **TransE** Bordes et al. (2013): Represents relations as translation vectors in the embedding space. For a triplet (h,r,t), it enforces $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$, where $\mathbf{h},\mathbf{r},\mathbf{t} \in \mathbb{R}^d$. TransE is intuitive and efficient and supports inverse and antisymmetric relations. However, it struggles with symmetric relations and complex compositions, especially for 1-to-N patterns. Conceptually, it is similar to the *word2vec* skip-gram model, which preserves semantic proximity via additive operations (e.g., king $-$ man $+$ woman $\approx$ queen).

- **TransR** Lin et al. (2015): Extends TransE by projecting entities into relation-specific vector spaces. Each relation r has a projection matrix $\mathbf{M}_r \in \mathbb{R}^{d_r \times d}$, which transforms entity embeddings $\mathbf{h},\mathbf{t} \in \mathbb{R}^d$ into the relational space. The scoring function becomes $\mathbf{M}_r\mathbf{h}+\mathbf{r} \approx \mathbf{M}_r\mathbf{t}$, where $\mathbf{r} \in \mathbb{R}^{d_r}$. This enables TransR to capture all relation types.

- **DistMult** Yang et al. (2015): Uses a bilinear scoring function $f(h,r,t) = \mathbf{h}^\top \mathrm{diag}(\mathbf{r})\mathbf{t}$, under the assumption of symmetric interactions. It performs well on symmetric relations and has limited support for 1-to-N patterns. However, it cannot capture inverse or antisymmetric properties due to the commutativity of multiplication. The scoring function can be interpreted as a similarity measure modulated by the relation vector, which is analogous to a weighted cosine similarity between the head and tail entities.

- **ComplEx** Trouillon et al. (2016): Extends DistMult into the complex number space. The scoring function is $\mathrm{Re}(\langle \mathbf{h},\mathbf{r},\bar{\mathbf{t}} \rangle)$, where $\bar{\mathbf{t}}$ is the complex conjugate of $\mathbf{t}$. This allows it to model symmetric, antisymmetric, and inverse relations effectively. However, it lacks mechanisms for representing transitive or compositional patterns.

The following table compares each model's scoring function, embedding space, and its ability to represent different relation types:

While Knowledge Graph embeddings enable the integration of structured relational information into machine learning applications, traditional recommendation approaches, whether based on collaborative filtering Koren et al. (2009) or deep learning Covington et al. (2016), typically

Model	Scoring Function	Space	Sym.	Antisym.	Inv.	Comp.	1-to-N
TransE	$-\|\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|\|$	$\mathbb{R}^d$		✓	✓	✓	
TransR	$-\|\|\mathbf{M}_r\mathbf{h}+\mathbf{r}-\mathbf{M}_r\mathbf{t}\|\|$	$\mathbb{R}^d, \mathbb{R}^{d_r}$	✓	✓	✓	✓	✓
DistMult	$\mathbf{h}^\top \mathrm{diag}(\mathbf{r})\mathbf{t}$	$\mathbb{R}^d$	✓				✓
ComplEx	$\mathrm{Re}(\langle \mathbf{h}, \mathbf{r}, \bar{\mathbf{t}} \rangle)$	$\mathbb{C}^d$	✓	✓	✓		✓

Table 2.2: Comparison of Knowledge Graph embedding models and their relation modeling capabilities.

operate within a static framework. In this framework, recommendations are generated independently, without considering the sequential nature of user interactions or the long-term impact of recommendation decisions Shani and Gunawardana (2011). This limitation has motivated the exploration of RL in RS Afsar et al. (2022).

2.3 Reinforcement Learning

Unlike supervised learning approaches, which optimize for immediate prediction accuracy, RL allows RS to learn optimal recommendation strategies considering both immediate feedback and long-term user engagement. This shift from static prediction to dynamic policy learning overcomes several limitations of traditional methods such as the cold-start problem, the exploration-exploitation trade-off, and the optimization of long-term user satisfaction.

2.3.1 Fundamentals of Reinforcement Learning

RL is a topic of machine learning in which an agent learns to make a sequence of decisions by engaging with an environment and receiving feedback in the form of rewards Sutton and Barto (2018). In contrast to supervised learning, where the learning process relies on labeled examples, RL requires the agent to explore and identify the sequence of actions that return the highest cumulative reward through trial and error without direct ground-truth labels after each action .

This process poses the challenge of temporal credit assignment Sutton (1984). The agent must determine which past actions led to eventual rewards or failures, since feedback may arrive long after critical decisions were made. The main focus of RL is to maximize the total reward over time, which involves learning a policy - or a strategy to act on the environment based on possible states - that ensures the highest return.

RL is grounded in the framework of a *Markov Decision Process* (MDP) Sutton and Barto (2018), which provides a formalism for modeling decision-making under uncertainty. An MDP is defined by a tuple (S, A, P, R, γ), where:

- S is the set of all possible states;
- A is the set of all possible actions;
- $P(s' \mid s, a)$ is the probability of transitioning to state $s' \in S$ from state $s \in S$ after taking action $a \in A$;

- $R(s,a)$ is the reward received after executing action a in state s;
- $\gamma \in 0,1$ is the discount factor, which weighs the importance of future rewards.

The agent aims to learn a policy $\pi^* : S \rightarrow A$ that maximizes the expected cumulative reward, also known as the return:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

A key challenge in RL is the balance between exploration and exploitation. Exploration refers to the agent trying new actions to discover potentially better rewards, while exploitation involves choosing actions that are already known to yield high rewards. Finding the right balance between these two behaviors is critical for effective learning, especially in environments where the agent has incomplete information or changing dynamics.

2.3.2 Evolution of RL Methods

Over the years, a variety of foundational algorithms have emerged in the domain of RL. One of the most significant value-based methods is Q-learning Watkins and Dayan (1992). Its objective is to learn the optimal action-value function by iteratively refining estimates based on the rewards and transitions observed in the environment. Alternatively, policy gradient methods adopt a distinct strategy by directly parameterizing the policy and modifying its parameters in order to enhance the expected reward. These methods have demonstrated considerable effectiveness in high-dimensional or continuous action spaces.

Deep RL has represented a milestone in this research area by merging the concepts of RL with the robust function approximation capabilities provided by deep neural networks. A big achievement in the field is the Deep Q-Network (DQN) Mnih et al. (2015), which showed that an agent could reach human-level performance in different Atari games by using a deep neural network to estimate the optimal Q-function. Based on this breakthrough, several new techniques have been developed, with actor-critic models Mnih et al. (2016) being particularly notable. These methods leverage the advantages of both value-based and policy-based approaches, resulting in further enhancements in performance and stability Schulman et al. (2017).

Value-based Methods Value-based methods focus on estimating the value function, which roughly calculates the expected return - or total reward - of executing an action in a particular state. The foundational concept of these approaches is the action-value function $Q(s,a)$, which indicates the expected long-term reward after performing action a in state s, following an optimal strategy.

Techniques like Q-learning refine the Q-values iteratively through the use of the Bellman equation, which helps distribute the observed rewards back through the state-action space. These methods are especially effective in settings with discrete action spaces, where it is feasible to calculate the maximum value among potential actions efficiently.

However, value-based methods can encounter difficulties in environments featuring continuous or high-dimensional action spaces and may sometimes face challenges such as overestimation bias. To address these issues and enhance performance, variations like Double Q-learning van Hasselt et al. (2015) have been introduced. Nevertheless, these methods can still suffer from high variance in Q-value estimates, instability during learning, and sensitivity to hyperparameters. In sparse reward environments, they may struggle to propagate meaningful signals, requiring mechanisms like reward shaping or target networks to stabilize updates.

Policy-based Methods Policy gradient methods Williams (1992) adopt a fundamentally distinct strategy by directly enhancing the strategy without the need for a value function. In these approaches, the policy ($\pi_\theta(a|s)$) is defined by a set of parameters (θ), and the goal is to modify these parameters to maximize the expected reward. This is usually achieved by calculating the gradient of the expected return with respect to the policy parameters and then updating them via gradient ascent.

A significant benefit of policy gradient methods is their capability to effectively manage continuous action spaces, as they produce a probability distribution over actions instead of a finite selection. Additionally, these methods often facilitate more stable learning when paired with techniques like baseline subtraction, which helps to lower the variance in the gradient estimates.

However, policy gradient methods also have notable limitations. They can be sampling inefficient due to high variance in gradient estimates and typically require more interactions with the environment. In addition, without appropriate exploration strategies, the learned policy may converge to suboptimal behavior.

2.3.3 RL-based Recommender Systems

Applying RL to RS requires a careful reformulation of the recommendation process within the MDP framework.

MDP Formulation for Recommendations In the context of RS, the MDP is formulated as the RS is an agent that learns an optimal recommendation policy by interacting with the environment, the users. At each time step, the agent observes the current user state, selects a recommendation action, and receives feedback in the form of rewards. This feedback influences the transition to the next state making the system consider the sequential nature of user interactions.

The key advantage of this formulation is its ability to model dynamic user behavior, where recommendations influence future preferences. This forces the system to adapt over the time to changes in the user behavior or the item availability. The temporal dependencies of the context and the cumulative effects of each recommendation are what guides RL-based RS.

State Representation In RL-based RS, the state representation is all the relevant information about the user's current context that may influence recommendation decisions. This information can include the user's interaction history, such as previously viewed, rated, or purchased items,

as well as temporal features, such as session duration and interaction frequency. A common deep learning-based approach is to encode each user interaction sequence as the state representation Afsar et al. (2022), with models such as LSTMs Hochreiter and Schmidhuber (1997) and GRUs Cho et al. (2014a).

It is in the state representation where Knowledge Graphs can deeply enhance RL-based RS, incorporating external information about every user-item interaction. For example, if a user has interacted with items belonging to specific genres, directors, or brands, the KG can provide context about related entities and their semantic relationships. This allows the state representation to capture implicit user preferences and interests that may not be directly observable from interaction history alone.

Action Space In the context of RS, the action space is defined as the set of all potential items that the agent is capable of recommending to a user during each interaction step. This can be formulated in a variety of ways. One possibility is to express it as a direct list recommendation, wherein the agent selects multiple items simultaneously. Another approach is to express it as individual item selection, where one item is chosen per interaction Zheng et al. (2018); Zhao et al. (2018).

The representation of actions is tied to the selection of the RL algorithm, being an important pillar for the learning process. Action representation can take several forms Chen et al. (2019b):

- **Discrete representation**: Each item is represented as a unique discrete action with a unique identifier, suitable for smaller catalogs but computationally challenging for large inventories;
- **Continuous representation**: Items are embedded in continuous vector spaces, enabling similarity-based reasoning Lillicrap et al. (2015);
- **Hybrid approaches**: Combine discrete and continuous representations to balance expressiveness with computational efficiency;

For systems with large item catalogs (e.g., millions of products), the action space becomes computationally intractable, requiring specialized techniques Chen et al. (2019a):

1. **Candidate generation**: Pre-filtering the action space to a subset of relevant items using collaborative filtering or content-based methods Covington et al. (2016);
2. **Hierarchical action spaces**: First selecting high-level categories or features, then choosing specific items within those categories Tang et al. (2019);
3. **Action clustering**: Grouping similar items and selecting clusters before individual items Ie et al. (2019);

Knowledge graphs can enhance action representation by incorporating rich semantic relationships between items, users, and contextual entities Xian et al. (2019b); Liu et al. (2020). This

enables agents to reason about item recommendations based on semantic similarity, entity relationships, and multi-hop connections in the KG, leading to more informed and explainable recommendation decisions.

Reward Design This is the most critical part of any RL-based system, as it determines directly the system optimization objective. In RS, it must balance the imediate user satisfaction with the long-term engagement.

Effective reward design often involves combining multiple signal types through weighted combinations or multi-objective optimization techniques. Some approaches use shaped rewards that provide intermediate feedback signals Chen et al. (2019c), while others use hierarchical reward structures that optimize for immediate and long-term objectives at different time scales Chen et al. (2019a). The challenge is effectively designing a reward functions that encourage long-term behavior while providing a sufficient learning signal for efficient policy optimization.

2.3.4 RL-based algorithms

Several RL algorithms have been adapted for use in RS. Offline RL methods have received particular attention because they can learn from logged user interaction data without requiring live experimentation. In some cases, simulators such as DIEN Zhou et al. (2019), LSTMs or Wide & Deep Cheng et al. (2016) are employed to simulate user responses and create controlled environments for policy evaluation Ie et al. (2019). Some of the foundational algorithms in Offline RL are BC, BCQ and CQL Afsar et al. (2022); Levine et al. (2020); Chen et al. (2021a).

Behavioral Cloning (BC) Pomerleau (1988) is the simplest approach to learning recommendation policies from historical data. In BC, the agent mimics the behavior in the logged dataset, treating the recommendation problem as a supervised learning problem. The policy $\pi(a|s)$ is trained to predict actions (recommendations) taken from the historical data given the corresponding states (user contexts).

Typically, this is done by minimizing the cross-entropy loss between the predicted and logged actions. Although BC is simple and stable, it suffers from distribution shift. During deployment, the learned policy may encounter states that are not well represented in the training data, causing the policy to extrapolate poorly. This distributional mismatch can lead to compounding errors, where small mistakes in initial recommendations cause the agent to enter regions of the state space that are not found during training. This further increases error as the trajectory progresses, a phenomenon known as error accumulation or covariate shift Ross et al. (2011).

Batch Constrained Q-learning (BCQ) Fujimoto et al. (2019) addresses the distributional shift problem seen in behavior cloning. BCQ accomplishes this by constraining the learned policy to select actions that are likely under the behavior policy distribution. This is done through a generative model that learns to produce actions similar to those in the dataset and a Q-function that evaluates these actions. Specifically, the agent samples candidate actions from the learned generative model, $G(a|s)$, and only considers actions with a high estimated likelihood under the behavior policy. These actions are filtered before being passed to the Q-function for evaluation.

In recommendation contexts, BCQ prevents the system from recommending items with small interaction history, thereby reducing the risk of poor performance due to extrapolation errors.

Conservative Q-Learning (CQL) Kumar et al. (2020) is a more conservative approach, directly addressing the overestimation bias inherent in offline Q-learning. It adds a regularization term to the Q-learning objective, penalizing Q-values for state-action pairs that are not present in the dataset and encourages higher Q-values for actions that appear in the logged data. This conservative approach ensures that the learned policy assigns lower values to out-of-sample actions. Thus, CQL is particularly suitable for RS, as recommending unseen or poorly supported items can lead to a poor user experience. CQL has demonstrated robust empirical performance in recommendation tasks Afsar et al. (2023), maintaining conservative estimates while enabling policy improvement over the behavior policy.

These offline RL algorithms are particularly valuable in RS because they can leverage the vast amounts of logged user interaction data that is commonly available in online platforms.

2.3.5 Available Toolkits

Several specialized toolkits have been developed to facilitate research and development in RL-based RS. Each toolkit addresses different aspects of the research pipeline, from simulation environments to real-world datasets.

RL4RS: Real-World Benchmark Dataset RL4RS Wang et al. (2021) is a open framework designed to close the gap between RL-based RS research and real-world applications. This framework is focused on slate recommendations, offering real-world datasets collected from real logged data from NetEase Game, providing a realistic sequential decision-making environment. This framework offer a wide variety of algorithms for various categories: supervised-learning, model-free RL and offline RL.

RecSim: A Configurable Recommender Systems Simulation Platform RecSim Ie et al. (2019) is a framework provided by Google Research as a contribution to simulation-based development of RL algorithms for RS. The biggest contribution of the framework is the possibility to create new environments that vary in assumptions about user preferences and item familiarity, user latent state dynamics, and choice models.

EasyRL4Rec EasyRL4Rec Yu et al. (2024) addresses the lack of user-friendly frameworks and inconsistent evaluation metrics. This comprehensive library provides lightweight and diverse RL environments based on five public datasets and includes core modules with rich options, simplifying model development while offering unified evaluation standards focusing on long-term outcomes. The library is structured around four core modules, each for each part of the MDP formulation of RL-based RS, allowing easier configuration of each step. The toolkit provides a variety of evaluation metrics commonly used in RL and RS. These metrics include cumulative

reward measures for assessing the long-term effects of RL policies and recommendation ranking for short-term user impact.

2.4 Related Work

This section reviews works that integrate KGs, RL, and RS to address recommendation challenges. Our focus is on methods that use all three components in a unified framework. We analyze their integration approaches, success metrics, and contributions to the field.

PGPR: Policy-Guided Path Reasoning PGPR Xian et al. (2019a) is a groundbreaking approach that couples recommendation and interpretability by providing actual paths in a KG. The method formulates the recommendation problem as a sequential decision process where the target user is defined as the initial state, the Knowledge Graph is the environment and each step on it is an action. The RL agent is then encouraged to find the best possible paths within the KG to recommend items. The output is which item was recommended and the connections it has to explain the recommendation. This type of KG integration effectively address the data sparsity and cold start problems, as if the item or user is in the KG, there will be a path inside it that leads to other entities.

However, PGPR has some limitations. The algorithm tends to favor shorter paths due to reward optimization, potentially overlooking longer paths that are more semantically meaningful. The approach also lacks consideration of path quality from an explanation perspective, focusing primarily on recommendation accuracy rather than explanation effectiveness.

In 2022, researchers extended the framework to use quality metrics of the reasoning paths, such as recency, popularity and diversity Balloccu et al. (2022). This made the RL agent not only optimize the recommendation quality, but also its explanation and meaningfulness.

KERL: A Knowledge-Guided Reinforcement Learning Model for Sequential Recommendation KERL Wang et al. (2020) is a framework that, instead of using the KG in a path reasoning approach, uses it to create better state and action representation, integrating directly the user history, item features and their contextual neighbors properties. With this, they approach the recommendation system as a sequential problem, where at each step the RL agent needs to recommend an item based on the current state, receiving the feedback and updating itself for the next step.

Specifically, KERL constructs three types of state representations that are concatenated to form the final state vector:

- **Sequence-level representation**: Captured by a GRU Hidasi et al. (2016b) over past interactions to encode sequential user behavior patterns;
- **Current knowledge-based preference**: Computed by averaging TransE KG embeddings of previously interacted items to represent existing user interests;

- **Future knowledge-based preference**: Predicted by a Multi-Layer Perceptron Rumelhart et al. (1986) (or induction network) applied to the current preference to anticipate possible shifts in user interests;

To encourage better recommendation sequence, KERL applies a composite reward function, where both the sequence creation and alignment of KG relationship are scored. They apply this setting for both next-item and next-session prediction. The datasets used in the article are primarily the Amazon He and McAuley (2016) and LastFM Schedl (2016). As baselines, we highlight GRU4Rec Hidasi et al. (2016a) and FPMC Rendle et al. (2010) as strong algorithms used in the article as comparison with sequential-based models.

Despite its innovations, KERL also faces some limitations, mostly in scalability when dealing with very large KGs due to the computational complexity of enhanced state representations. Additionally, the composite reward function requires careful tuning of the balance between sequence-level and knowledge-level rewards, which may not generalize well across different domains and be an intensive fine tuning work. Lastly, the method relies on pre-computed KG embeddings, which may limit its ability to adapt to evolving graphs over the time.

Interactive Recommender System via Knowledge Graph-enhanced Reinforcement Learning Knowledge Graphs are also being used to solve sample efficiency in interactive RS, where the users preferences are highly dynamic. Using the KG as prior knowledge about the items, the recommendation steps are guided by the semantic properties of the items and the previous user interactions Zhou et al. (2020). The action space is refined using the KG for candidate generation, focusing on the exploration of items that are semantically connected to the user. This ensures that most of the recommendations are relevant overall.

The performance of the method depends heavily on the quality of the KG, which can be incomplete in several cases or contain biased information. Additionally, the approach requires the careful design of the candidate selection mechanisms, which may not transfer well across different recommendation domains. Furthermore, the method's emphasis on sample efficiency could harm the diversity of recommendations.

Temporal Graph Contrastive Learning for Sequential Recommendation TGCL4SR Zhang et al. (2024) attempts to solve the problem of exploiting temporal information while dealing with uncertainty and noise in evolving user behaviors for next-item prediction. The proposed method uses a Temporal Graph Contrastive Learning approach, leveraging local interaction sequences and global temporal graphs to understand the correlation between items and analyze user behavior temporally.

This approach enhances the temporal graph by applying two types of transformations: one that samples neighboring nodes and another that slightly modifies time information. These methods create different versions of the same graph sequence. Then, a Temporal Item Transition Graph Convolutional Network (TiTConv) is used to learn patterns in how items change over time, producing a vector representation of it. To make it more consistent, Temporal Graph Contrastive Learning

(TGCL) encourages the model to produce similar representations for the same sequences across these variations.

Despite being very innovative, TGCL4SR has a high computational cost due to process both global relationship and local sequences. As it is dependent on a two transformations, there are a lot of parameters to tune, which may not generalize well.

Key trends in latest works The recent works that combine the areas of RL, Representation Learning, more specifically with KG, and RS have a clear objective of trying to solve problems seen on traditional RL-based RS approaches, such as sample efficiency and temporal dynamics.

Knowledge Graphs are being used heavily in state representation, where the addition if semantic properties between items and users in the process has shown big improvements in the quality of recommendations. Nevertheless, KGs are being successfully used to filter the candidates before the action, improving both scalability and efficiency of these systems. In the case of explainable RS, KGs are a clear benefit in path reasoning algorithms, allowing the user to know exactly what the system saw to recommend that specific item.

The community is also focusing on ways to model the evolution of the graphs, as previously graphs were taken as static structures. This represented a big weakness of the approaches, as they relied on the exploration capability of RL. The embeddings are now being time-aware and the rewards functions are now taking account the temporal consistency of the user behavior patterns.

The field is also evolving for more comprehensive evaluation frameworks, not just focused on recommendation metrics, but also explainability, diversity and semantic alignment with the Knowledge Graphs. This indicates that the field is not only focusing on pure optimization, but also on improving user experience in recommendation platforms.

Positioning of This Work This study builds upon the foundations established by KERL and analogous KG-enhanced RL approaches, while addressing key limitations and extending the methodology in several significant directions. In contrast to KERL's intricate three-component state representation, which incorporates future preference prediction through induction networks, our approach employs a more streamlined methodology that emphasizes direct integration of TransE embeddings into behavioral cloning frameworks.

Specifically, our work differs from existing approaches in three key aspects. First, we use behavorial cloning in the place of full RL. This approach simplifies the learning process while maintaining the benefits of sequential decision-making frameworks. This choice facilitates more stable training dynamics and easier interpretability compared to the complex reward functions required by methods like KERL. Secondly, we conduct systematic ablation studies across multiple user representation strategies to ascertain the most effective semantic integration approaches for sequential recommendation. Lastly, we compare these approaches to non-KG statistical baselines, making possible to quantify the contribution of semantic information in state representation.

This work contributes to the field by providing a controlled evaluation environment that isolates the impact of KG integration from other modeling choices. By maintaining identical temporal splits, session boundaries, and evaluation protocols across all approaches, our methodology enables fair comparison between semantic-enhanced and traditional sequential recommendation methods. This systematic approach addresses a gap in existing literature where performance improvements are often obfuscated by multiple simultaneous changes to model architecture, data processing, and evaluation methodology.

Chapter 3

Methodology

This chapter presents the theoretical framework for integrating KG embeddings into sequential recommendation systems. We establish the methodological foundations that differentiate this approach from traditional methods, describe the system architecture, and detail the core processes for knowledge integration and RL-based recommendation.

3.1 Problem Formalization

We formalize the KG-enhanced sequential recommendation as an extension of the standard Markov Decision Process framework. Let $\phi : E \to \mathbb{R}^d$ denote the KG embedding function that maps entities to d-dimensional vectors.

The enhanced MDP is defined as the tuple $(S_{KG}, A, P, R, \gamma)$ where:

- $S_{KG} = \{f_{embed}(\{\phi(e) \mid e \in H_u\}, \phi(i_t)) \mid u \in U, t \in \mathbb{N}\}$ is the set of KG-enhanced states;
- $A = \{a_i \mid i \in I\}$ is the set of recommendation actions (items);
- $P(s'_{KG} \mid s_{KG}, a)$ is the transition probability to enhanced state s'_{KG} from state s_{KG} after action a;
- $R(s_{KG}, a)$ is the reward function mapping enhanced states and actions to user satisfaction signals;
- $\gamma \in [0, 1]$ is the discount factor for future rewards.

where f_{embed} represents a state construction function, H_u denotes user interaction history, i_t is the latest item context, U is the user set, and I is the item set.

3.2 System Architecture

Figure 3.1 presents the theoretical architecture for KG-enhanced sequential recommendation. The framework operates through four distinct processing stages that transform heterogeneous data sources into knowledge-guided recommendations.

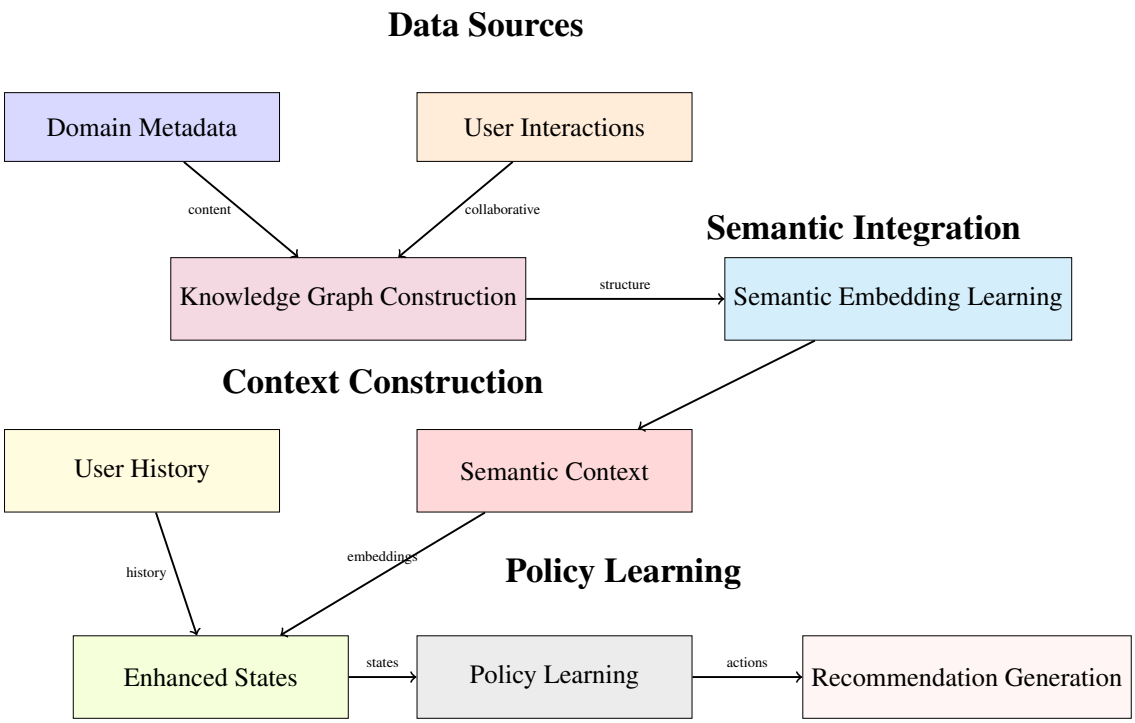

Figure 3.1: Architecture of the developed system.

The system integrates multiple information modalities through a unified processing pipeline. We will illustrate the process using a running example from an e-commerce platform in which a user browses electronics products.

Data Sources This layer combines two complementary information sources. *Domain metadata* provides structured knowledge about items, including attributes, taxonomies, and relationships. For instance, it includes product categories ("smartphones," "accessories"), specifications (screen size, battery capacity), and brand information. *User interactions* capture behavioral signals, such as clicks, purchases, and ratings. In our example, a user's session might include viewing a cellphone from a specific brand, clicking on wireless earbuds, and purchasing a phone case. Systematic integration of these heterogeneous sources is required to create coherent knowledge representations that bridge content features with collaborative patterns

Semantic Integration This stage processes integrated data through two complementary mechanisms. *Knowledge Graph Construction* normalizes domain metadata and user preferences into structured relationships, creating a unified graph in which entities (e.g., users, items, and attributes) are connected by typed edges (e.g., "belongs-to" "purchased," and "has-feature"). In our example, this creates connections such as

$$\texttt{Samsung\_Phone} \xrightarrow{\text{category}} \texttt{Smartphones}$$

$$\texttt{Wireless\_Earbuds} \xrightarrow{\text{frequently-bought-with}} \texttt{Samsung\_Phone}$$

Semantic Embedding Learning then maps these graph entities and relations into dense vector representations, enabling the computation of semantic similarity. Items with similar features or co-purchase patterns receive similar embeddings. This allows the system to recognize that a phone case is semantically relevant to a recently viewed phone, even without explicit interaction data.

Context Construction Enhanced state representations are constructed by fusing both *User history Semantic context*. The first captures behavioral patterns from the interaction sequence. In our example, this would be the temporal progression from viewing phones to exploring accessories. The second enriches this behavioral signal with knowledge-based features derived from learned embeddings. These features include category information (e.g. electronics $\rightarrow$ mobile accessories) and attribute similarities (e.g. compatible devices). This dual-context approach allows the system to understand both what the user has done and why certain items are related, creating representations that capture temporal dynamics and conceptual connections.

Policy Learning Sequential decision-making uses enhanced state representations within a RL framework. At each step, the RL agent observes the current state by combining user history and semantic context, selects an action by recommending an item, and receives feedback in the form of rewards, or user engagement signals. In our example, after viewing a Samsung phone and clicking on earbuds, the policy network uses the interaction sequence and semantic knowledge to recommend accessories like phone cases or screen protectors. Offline RL algorithms learn from logged trajectories without requiring online exploration. This enables the system to generate personalized recommendations that optimize long-term user engagement while respecting the semantic structure of the item space.

3.3 Semantic State Construction

3.3.1 Knowledge Graph Embedding Framework

The semantic enhancement process begins with learning dense vector representations of KG entities. Let $G = (E, R, T)$ represent the KG where $T \subseteq E \times R \times E$ contains the relational triples. The embedding learning objective employs a pairwise ranking loss:

$$\mathscr{L}_{KG} = \sum_{(h,r,t)\in T} \sum_{(h',r,t')\in T_{neg}} L_{rank}(f(h,r,t), f(h',r,t'))$$

where $\phi : E \rightarrow \mathbb{R}^d$ maps entities to d-dimensional vectors, $\psi : R \rightarrow \mathbb{R}^d$ maps relations to the same space, $f(h,r,t)$ is a model-specific scoring function, T_{neg} contains negative triples generated through corruption and L_{rank} represents a ranking loss function.

The learned embeddings should satisfy the property that semantically related entities have similar vector representations, while unrelated entities are represented separately in the embedding space.

3.3.2 State Enhancement Integration

The enhanced state at time t integrates user behavioral patterns with semantic item characteristics:

$$
\begin{aligned}
s_{enhanced}^{(t)} &= [s_{user}^{(t)}; s_{item}^{(t)}] \\
s_{user}^{(t)} &= \mathrm{Agg}_{\theta}\left(\{\phi(e_i) \mid e_i \in H_u^{(t)}\}\right) \\
s_{item}^{(t)} &= \phi(e_{current})
\end{aligned}
$$

where $H_u^{(t)} = \{e_1, e_2, \ldots, e_k\}$ represents the user's interaction history up to time t, $e_{current}$ denotes the current item entity, and the concatenation operation creates the unified state representation.

The user component, $s_{user}^{(t)}$, captures preference patterns through the use of aggregated semantic representations of items with which the user has previously interacted. Meanwhile, $s_{item}^{(t)}$ provides immediate context about the semantic characteristics of the current item. Figure 3.2 illustrates this process, with each step of the pipeline explained below.

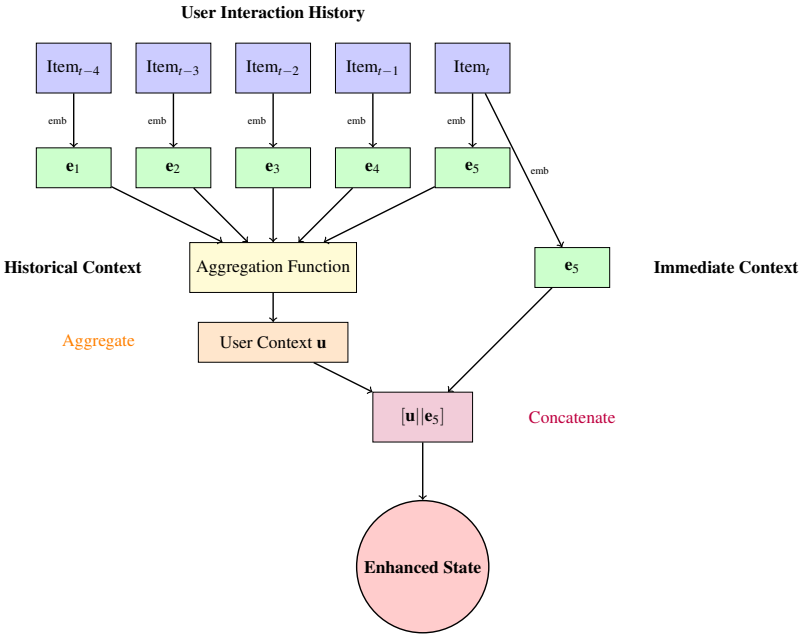

Figure 3.2: Enhanced state construction process.

User Context Aggregation User representations are constructed by aggregating item embeddings from recent interactions. This process transforms sequential user behavior into semantic vector representations that capture preference patterns based on content relationships.

Most Recent Item Representation The item with which the user most recently interacted provides immediate semantic context for the current decision point. This representation encodes the item's content and its relationships with other entities in the KG. It serves as contextual information for predicting the user's next interaction.

State Vector Construction Enhanced states are created by concatenating the aggregated user context vector with the embedding of the most recently interacted item. This creates a unified input representation, allowing the policy to leverage the user's historical preference patterns and the current context's semantic characteristics to predict the next item in the sequence.

Aggregation Functions The aggregation function Agg_θ determines how historical embeddings are combined to form user representations. We define a parameterized weighting scheme:

$$\text{Agg}_\theta(H_u^{(t)}) = \sum_{i=1}^{k} w_i(\theta, t, i) \cdot \phi(e_i) \tag{3.1}$$

where $w_i(\theta, t, i)$ represents context-dependent weights that sum to unity: $\sum_{i=1}^{k} w_i = 1$.

Mean aggregation : Computes the simple average of recent interaction embeddings within the embeddings space, treating all historical interactions equally to establish a baseline user representation:

$$w_i = \frac{1}{k}, \quad \forall i \in \{1, 2, \ldots, k\}$$

Recency-based : Applies temporal weighting to prioritize more recent interactions in the embedding space. This reflects the dynamic nature of user preferences and ensures that current behavior patterns have a greater influence on the representation:

$$w_i = \frac{i}{\sum_{j=1}^{k} j}$$

where i represents the position of interaction e_i in the chronologically ordered history (with $i = 1$ for oldest, $i = k$ for newest).

Quality-based : Weighted aggregation based on user satisfaction signals. This ensures that positive experiences contribute more significantly to the user representation than neutral or negative ones;

$$w_i = \frac{f_{quality}(r_i)}{\sum_{j=1}^{k} f_{quality}(r_j)}$$

where r_i is the binary reward associated with interaction e_i, and the quality function assigns weights based on user satisfaction:

$$f_{quality}(r) = \begin{cases} \beta_{pos} & \text{if } r = 1 \\ \beta_{neg} & \text{if } r = 0 \end{cases} \tag{3.2}$$

where $\beta_{pos} > \beta_{neg} > 0$ control the level importance of positive versus negative feedback interactions.

3.4 Reinforcement Learning Integration

Enhanced semantic state representations enable policy learning through offline RL. Unlike the online setting, which relies on active user interaction, offline RL operates exclusively on fixed logged data, eliminating the need for environmental exploration.

The Distributional Shift Challenge The central challenge in offline RL is distributional shift, where the learned policy π may select actions that are not well-represented in the logged dataset $\mathscr{D}$, which was collected under a potentially different behavioral policy π_β. When the policy encounters states or state-action pairs outside the data distribution, value estimates become unreliable, which can lead to policy degradation.

Semantic state enhancement partially mitigates this challenge by enabling better generalization through KG-based similarities. When a policy encounters a new state-action pair, the semantic embeddings enable the algorithm to leverage information from similar items with observed interactions, thereby improving extrapolation beyond the data's scope.

3.4.1 Policy Learning with Enhanced States

To address the distributional shift challenge, we limit the scope of this study into three offline RL algorithm categories known for handling it:

Behavioral Imitation This method directly imitates the logging policy via supervised learning, rather than attempting value-based improvement. By replicating observed behaviors, this approach avoids Q-value overestimation errors arising from distributional shift. However, it is limited by the quality of the behavioral policy and cannot surpass the performance observed in the training data.

Policy Constraint Methods These methods perform value-based learning while explicitly constraining the learned policy to stay close to the behavioral policy's action distribution. This balances policy improvement with safety by limiting exploration to regions that are well covered by the dataset.

Conservative Value Estimation It allows for arbitrary policy actions, but it regularizes value functions to be pessimistic for out-of-distribution state-action pairs. This prevents overestimation errors from propagating through the learning process.

3.4.2 Integration with Semantic Enhancement

The KG-enhanced state construction integrates semantic information into the offline RL framework through the concatenated state representation $s^{(t)}_{enhanced} = [s^{(t)}_{user}; s^{(t)}_{item}]$, as defined in Section 3.3.

Implicit Semantic Generalization Incorporating KG embeddings into the state representation gives the neural network semantic features to complement behavioral signals. Because of the continuity properties of neural network function approximators, items with similar embeddings will tend to receive similar policy outputs or Q-values, provided the learned function assigns weight to these semantic features.

Enhanced State Space Semantic embeddings augment behavioral features with content-based information, effectively expanding state representations beyond purely collaborative signals. This enriched representation gives offline RL algorithms additional context for making decisions, especially for items with limited interaction history where collaborative signals are scarce.

Architectural Implementation The KG embeddings $\phi(e)$ are precomputed and remain fixed during RL training. The policy network, π_θ, processes the enhanced states through its hidden layers and learns to combine behavioral and semantic features for action selection. This modular design enables the exploration of various aggregation strategies, such as the ones described in Section 3.3, while ensuring consistent semantic representations across all tested offline RL algorithms.

Chapter 4

Experimental Setup

This chapter explains our study's implementation details. We present the dataset selection, its statistics, the details of the KG construction, model architectures, and the training procedures.

4.1 Dataset and Preprocessing

This section details the data preparation workflow, with Figure 4.1 illustrating the complete processing pipeline used to prepare data for model training and evaluation.

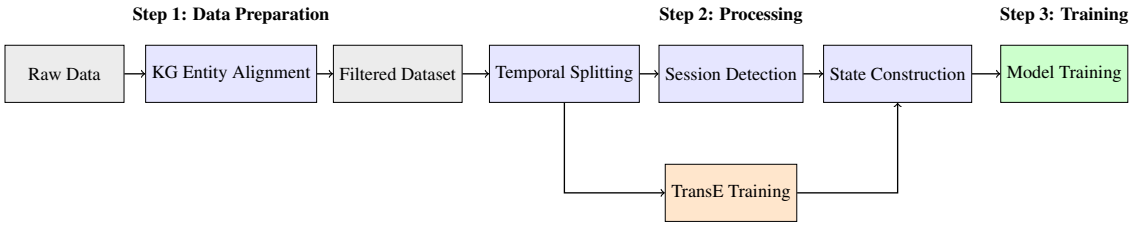

Figure 4.1: Data processing pipeline with parallel knowledge graph embedding training.

Dataset Selection For this research, the MovieLens dataset Harper and Konstan (2015) was chosen due to its wide adoption within the RS research community. This dataset collection aggregates variants from 100,000 to 32 million user ratings, the majority having corresponded linkage to Knowledge Graphs provided by open source communities, such as DBpedia Auer et al. (2007).

Given the high computational cost and the limitations in processing power, this study employs the Movielens-1M variant. This size combines both computational effiency and experimental validity, maintaining sufficient scale to show statistically significant results while remaining tractable for extensive experimentation.

Original MovieLens-1M Properties The MovieLens-1M dataset contains comprehensive user-item interaction data spanning approximately three years of user activity. Table 4.1 shows the basic information about the raw dataset.

Property	Value
Total ratings	1,000,209
Unique users	6,040
Unique items	3,706
Density	4.47%
Rating range	1–5
Average rating	3.582
Date range	Apr 25, 2000 – Feb 28, 2003

Table 4.1: Original MovieLens-1M dataset statistics before preprocessing.

The dataset exhibits characteristics that are typical of real-world recommendation scenarios. With a density of 4.47% (percentage of all possible user-item pairs have ratings), it shows the sparse interaction patterns that are common in practical systems. The three-year temporal span provides sufficient data for learning sequential patterns, and the rating scale enables implicit and explicit feedback. High-quality KG mappings were available for the majority of items, which was a decisive factor in dataset selection.

Knowledge Graph Filtering Impact In order to ensure consistency between rating data and KG coverage, entity filtering was applied to remove items without corresponding DBpedia representations. These items had fewer interactions in the original data, causing a small increase in dataset density. As shown in Table 4.2, this preprocessing step removes a considerable amount of data from the initial composition of the dataset.

Property	Before	After
Total ratings	1,000,209	948,988
Unique users	6,040	6,040
Unique items	3,706	3,228
Density	4.47%	4.87%
Average rating	3.582	3.575
Filtering Impact		
Removed items	478 (12.9%)	
Removed ratings	51,221 (5.1%)	
Retention rate	94.9%	

Table 4.2: Impact of KG entity alignment on dataset composition.

Temporal Split Characteristics In accordance with the best practices for evaluating sequential recommendations, the dataset was partitioned chronologically using 80/10/10 timestamp quantiles. Table 1 presents the resulting split characteristics.

The temporal split maintains the fundamental data distribution across the training, validation, and test sets, exhibiting comparable density and item coverage throughout all sets. The temporal split based on a global timeline ensures no leakage of information between sets.

Property	Training	Validation	Test
Date cutoff	$\leq$ Dec 2, 2000	Dec 2–29, 2000	$>$ Dec 29, 2000
Total ratings	759,190	94,899	94,899
Unique users	5,399	1,124	1,207
Unique items	3,199	2,980	3,053
Density	4.40%	2.83%	2.58%
Cold-start Characteristics			
Cold users	–	608 (54.1%)	202 (16.7%)
Cold items	–	12 (0.4%)	21 (0.7%)

Table 4.3: Temporal split statistics and cold-start analysis.

It is important to note that the temporal split caused a cold-start effect for users. Over half of the validation users and 16.7% of the test users are entirely new to the model at evaluation time. However, cold-start items are minimal due to the comprehensive item coverage in training. This setup simulates the real-world challenge of recommending to new or infrequent users without introducing artificial changes in item or interaction distributions.

Session Boundary Detection Sequential recommendation requires well-defined session boundaries to segment continuous user interactions into meaningful episodes. We implement a hybrid session detection approach:

- **Time-based segmentation**: Sessions terminate when the gap between consecutive interactions exceeds 1 hour (3,600 seconds);
- **User transition detection**: Session boundaries occur at user changes;
- **Hybrid integration**: Final boundaries determined by the union of both criteria;

This hybrid approach ensures sessions capture coherent user intentions while respecting natural temporal boundaries and preventing cross-user contamination. The 1-hour threshold was chosen based on typical movie browsing session lengths and aligns with conventions in session-based recommendation literature.

4.2 Knowledge Graph

4.2.1 Knowledge Graph Architecture

Our KG construction methodology integrates structured DBpedia Auer et al. (2007) metadata with collaborative signals by incorporating user entities and preference relationships directly into the graph structure. We formalize positive user feedback as explicit triples, thereby creating a hybrid knowledge representation that combines content-based semantic relationships with behavioral user-item interactions.

Collaborative Knowledge Integration We define positive user feedback as ratings of 4 or above on the 5-point scale. This results in 437,309 user-item preference relationships from the training set. This threshold balances signal quality with coverage, capturing clear positive user preferences while maintaining sufficient collaborative signal density.

Collaborative filtering relationships are formalized as KG triples of the form $(u_i, \text{interacted}, m_j)$, where each triple represents a positive preference relationship between a user entity and an item entity within the unified KG structure. This formulation enables the integration of explicit user preferences alongside content-based relationships, allowing TransE to learn unified embeddings that simultaneously encode content similarity patterns (e.g. genre and director relationships) and collaborative filtering signals (e.g. user preference clusters and item co-occurrence patterns). The resulting embedding space captures both the semantic properties of items through their metadata relationships and the behavioral patterns of users through their interaction histories.

Entity and Relation Statistics Table 4.4 presents the overall statistics of the constructed KG after integration of all information sources.

Component	Count
Entities	
Total entities	35,493
Movies	3,300
Users	6,040
Other entities	26,153
Relations	
Total relation types	42
Triples	
Content triples (DBpedia)	434,190
Collaborative triples (ratings ≥ 4)	437,309
Total triples	871,499

Table 4.4: Knowledge graph composition and statistics

The KG integrates 434,190 content-based triples from DBpedia with 437,309 collaborative triples derived from positive user ratings (rating ≥ 4), creating a balanced representation of both semantic relationships and user preferences.

4.2.2 TransE Embedding Training Configuration

The selection of TransE as the KG embedding method for this initial approach was motivated by its conceptual simplicity and computational efficiency. TransE operates within a single embedding space, wherein all entities are represented as vectors in the same dimensional space, and relationships are modeled as translation operations. This unified representation facilitates straightforward integration with recommendation models, as opposed to more complex embedding methods that may require relation-specific projections or complex number operations, such as TransR

and ComplEx, as seen in Section 2.2.3. While more sophisticated approaches might capture certain relational patterns more effectively, TransE is solid foundation in the scope of this study. The configuration details are presented in Table 4.5.

Parameter	Value
Embedding dimension	48
Training epochs	100
Batch size	256
Learning rate	0.001
Optimizer	Adam
Negative sampling ratio	1:1
Early stopping patience	3
Loss function	Margin-based ranking

Table 4.5: TransE training configuration and hyperparameters

The embedding dimension of 48 is selected to balance representational capacity with computational efficiency, while the training configuration follows established best practices for KG embedding with early stopping to prevent overfitting. The framework used for TransE training was PyKEEN 1.11.1 [Ali et al. (2021)] in an Apple M3 Pro CPU.

4.3 Model Architectures and Training Procedures

This section presents the implementation details and training configurations for the three offline reinforcement learning algorithms evaluated in this study: Behavioral Cloning (BC), Batch Constrained Q-Learning (BCQ), and Conservative Q-Learning (CQL). While all models share a common neural network architecture, they differ in their training objectives and algorithmic approaches. Specific details about each model can be seen in Section 2.3.4.

4.3.1 Common Encoder Architecture

All algorithms share the same encoder architecture to ensure fair comparison across methods, with parameters shown in Table 4.6.

Component	Specification	Output Dimension
Input layer	Enhanced state vector	96
Hidden layer 1	512 units, ReLU, BatchNorm	512
Hidden layer 2	512 units, ReLU, BatchNorm	512
Hidden layer 3	256 units, ReLU, BatchNorm	256
Dropout layer	Rate = 0.1	256
Output layer	Algorithm-specific	3,199

Table 4.6: Common neural network architecture across all algorithms

The architecture incorporates batch normalization after each hidden layer to stabilize the training process and improve convergence properties. Dropout regularization with a rate of 0.1 is applied to prevent overfitting. The input dimension of 96 corresponds to the concatenated TransE embeddings, and the output dimension of 3,199 matches the size of the action space derived from the 0-indexed training catalog.

4.3.2 Behavioral Cloning Implementation

The BC model utilizes cross-entropy loss as the objective function. Training parameters are presented in Table 4.7.

Parameter	**Value**
Framework	d3rlpy 2.0+
Loss function	Cross-entropy
Optimizer	Adam
Learning rate	0.0003
Batch size	1,024
Training steps	100,000
Weight decay	0.0
Gradient clipping	10.0

Table 4.7: Behavioral cloning training configuration.

Initial experiments revealed that the Adam optimizer with learning rate 0.0003 provided the most stable training dynamics across different state representation approaches. The batch size and training steps balance computational efficiency with sufficient exposure to training data.

4.3.3 Batch Constrained Q-Learning Implementation

BCQ uses a lower learning rate than BC does to account for the added complexity of Q-function learning. The algorithm-specific hyperparameters are detailed in Table 4.8.

Parameter	Value
Framework	d3rlpy 2.0+
Optimizer	Adam
Learning rate	6.25e-05
Batch size	1,024
Training steps	100,000
Discount factor (γ)	0.99
Weight decay	0.0
Gradient clipping	10.0
Action flexibility (τ)	0.3
BC–Q balance (β)	0.5
Target update interval	8,000
Number of critics	1
Share encoder	True

Table 4.8: BCQ training configuration.

The action flexibility parameter ($\tau = 0.3$) establishes the threshold for filtering actions, and the BC-Q balance parameter ($\beta = 0.5$) adjusts the weight of the combination of Q-learning and behavioral cloning losses. The shared encoder architecture reduces model complexity while maintaining expressive capacity.

4.3.4 Conservative Q-Learning Implementation

CQL shares similar base hyperparameters with BCQ but introduces the conservative regularization mechanism. The configuration parameters are presented in Table 4.9.

Parameter	Value
Framework	d3rlpy 2.0+
Optimizer	Adam
Learning rate	6.25e-05
Batch size	1,024
Training steps	100,000
Discount factor (γ)	0.99
Weight decay	0.0
Gradient clipping	10.0
CQL regularization (α)	2.0
Target update interval	8,000
Number of critics	1

Table 4.9: CQL training configuration.

The CQL regularization weight, $\alpha = 2.0$, controls the strength of the conservative penalty applied to out-of-distribution actions. Shared encoder is not implemented for discrete CQL in d3rlpy.

4.3.5 GRU4Rec Implementation

We utilized the PyTorch official GRU4Rec implementation. The baseline configuration is specified in Table 4.10.

Parameter	Value
Hidden layers	[100]
Loss function	BPR-max
Batch size	50
Learning rate	0.05
Training epochs	10
Dropout (hidden)	0.5
Momentum	0.0
BPR regularization	1.0
LogQ regularization	0.0
Sampling parameter	0.5
Negative samples	2,048

Table 4.10: GRU4Rec baseline training configuration.

The configuration follows the official recommendations for session-based recommendation with BPR-max loss, which has shown better performance compared to cross-entropy loss in sequential recommendation scenarios.

4.3.6 State Representation Methodologies

We implement two distinct state construction methodologies to evaluate the contribution of KG information in behavioral cloning for sequential recommendation.

Knowledge Graph-Enhanced States All KG-enhanced approaches use TransE embeddings with a context window of 5 recent items, concatenated with the current item embedding, yielding 96-dimensional state vectors:

$$s_t = [h_{\text{user}}(t); e_{\text{item}}(t)] \in \mathbb{R}^{96}$$

Mean Aggregation Simple mean averaging of recent item embeddings:

$$h_{\text{user}}(t) = \frac{1}{k} \sum_{i=1}^{k} \phi(e_i)$$

Temporal Weighting Linear progressive weights emphasizing recent interactions. For 5-item history, weights are [1, 2, 3, 4, 5] (normalized), with position 5 being most recent:

$$w_i = \frac{i}{\sum_{j=1}^{k} j}$$

Quality-based Weighting Differential weights based on user satisfaction:

- Positive feedback (rating ≥ 4): weight = 3.0
- Negative/neutral feedback (rating < 4): weight = 1.0

Weights are normalized to sum to 1 before computing the weighted average.

Non-KG Statistical States As a baseline comparison, we implement the statistical feature-based approach. User representation is composed by 6 aggregated statistics over the last 5 interactions: interaction count, mean reward, maximum reward, minimum reward, reward standard deviation, and average popularity of interacted items. Current item features comprise 4 characteristics: average reward , popularity, log-scaled interaction count, and user coverage. The resulting 10-dimensional state vector: $s_t \in \mathbb{R}^{10}$.

Item features are computed once from the entire training dataset before training and remain static throughout the process. History features, in contrast, are dynamically updated at each step based on the user's recent interaction sequence as they progress through their session. The computation of all item characteristics is derived from training data statistics and consistently applied across validation and test sets to prevent data leakage. This process was done to ensure similar treatment with both statistical and KG-based approaches.

For both settings, zero-padding is applied when users have fewer than 5 historical interactions to maintain consistent dimensionality.

4.4 Evaluation

This section presents the evaluation methodology used to assess all models on identical prediction tasks, ensuring a fair comparison of different algorithmic approaches. We describe the session-based ranking protocol, define the evaluation metrics, and explain the steps taken to ensure consistent experimentation.

4.4.1 Session-based Ranking Evaluation

All models are evaluated using a unified next-item prediction protocol, which accommodates both the state-action paradigm of RL methods and the sequential processing of GRU4Rec. The Figure 4.2 illustrates the whole process.

Prediction Task Generation For each user session in the test set:

- **Input**: Session history up to time step t;
- **Task**: Rank all items in the catalog for prediction at $t+1$;
- **Ground truth**: The actual item interacted with at $t+1$;

Multiple prediction tasks are generated per session. Each timestep (except the first) serves as an opportunity for prediction. This maximizes the amount of evaluation data while maintaining temporal integrity.

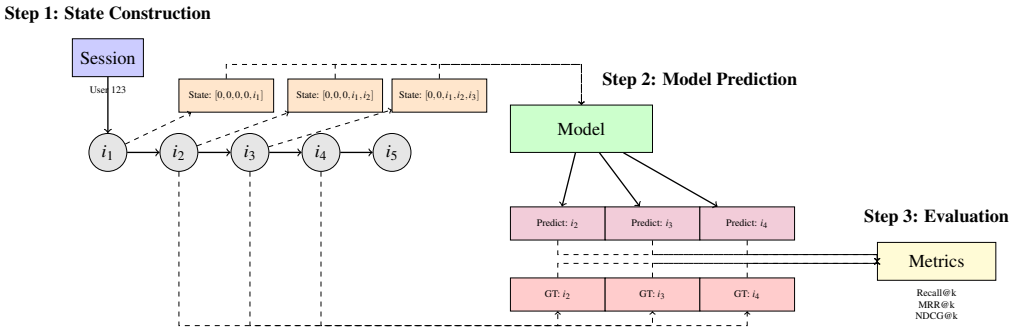

Figure 4.2: Session-based evaluation protocol.

Model-Specific Processing

- **RL models**: Convert session history into enhanced states using a 5-item context window with zero-padding for shorter histories. The learned policy generates action probabilities (for BC) or Q-values (for BCQ/CQL) for ranking.
- **GRU4Rec**: Process the full session prefix through the recurrent architecture to generate next-item predictions.

4.4.2 Evaluation Metrics

We employ three standard ranking metrics computed at cutoff values $k \in \{1, 5, 10, 20\}$:

Recall@k Proportion of test cases where the true next item appears in the top-k predictions:

$$\text{Recall@k} = \frac{1}{|Q|} \sum_{q \in Q} \begin{cases} 1 & \text{if } \text{rank}(i_q) \leq k \\ 0 & \text{otherwise} \end{cases}$$

MRR@k (Mean Reciprocal Rank) Average inverse rank of the correct item, considering only top-k:

$$\text{MRR@k} = \frac{1}{|Q|} \sum_{q \in Q} \begin{cases} \frac{1}{\text{rank}(i_q)} & \text{if } \text{rank}(i_q) \leq k \\ 0 & \text{otherwise} \end{cases}$$

NDCG@k (Normalized Discounted Cumulative Gain) Ranking quality metric with logarithmic position discounting:

$$\text{NDCG@k} = \frac{1}{|Q|} \sum_{q \in Q} \begin{cases} \frac{1}{\log_2(\text{rank}(i_q)+1)} & \text{if } \text{rank}(i_q) \leq k \\ 0 & \text{otherwise} \end{cases}$$

where Q represents all prediction tasks generated from the evaluation sessions, i_q is the ground-truth item for task q, and $\text{rank}(i_q)$ denotes the position of the correct item in the ranked list.

Chapter 5

Results and Analysis

This chapter presents the results of our study comparing knowledge graph-enhanced behavioral cloning and GRU4Rec for sequential recommendation.

5.1 Overall Performance

As illustrated in Table 5.1, we compare the GRU4Rec baseline against BC, BCQ, and CQL variants using the metrics described in Section 4.4. Four BC variants are evaluated: **BC-Mean** employs mean aggregation of TransE embeddings, **BC-Recency** uses recency-weighted TransE embeddings, **BC-Quality** applies quality-weighted TransE embeddings, and **BC-Stats** implements the non-KG statistical baseline. We mirror these for BCQ (**BCQ-Mean/Recency/Quality/Stats**) and CQL (**CQL-Mean/Recency/Quality/Stats**) as well.

	Recall@k				**MRR@k**				**NDCG@k**		
Method	**1**	**5**	**10**	**20**	**1**	**5**	**10**	**20**	**5**	**10**	**20**
GRU4Rec	.0223	.1034	.1844	.2998	.0223	.0489	.0595	.0674	.0623	.0883	.1173
BC-Mean	.0279	.1060	.1787	.2783	.0279	.0539	.0634	.0702	.0667	.0901	.1151
BC-Recency	.0274	.1036	.1718	.2691	.0274	.0528	.0617	.0684	.0653	.0872	.1116
BC-Quality	.0284	.1063	.1779	.2782	.0284	.0544	.0638	.0706	.0672	.0901	.1154
BC-Stats	.0138	.0537	.0912	.1476	.0138	.0269	.0319	.0357	.0335	.0455	.0597
BCQ-Mean	**.0319**	**.1198**	**.1968**	**.3052**	**.0319**	**.0613**	**.0714**	**.0788**	**.0757**	**.1004**	**.1276**
BCQ-Recency	**.0333**	**.1203**	**.1973**	**.3072**	**.0333**	**.0624**	**.0725**	**.0801**	**.0767**	**.1014**	**.1291**
BCQ-Quality	**.0320**	**.1201**	**.1970**	**.3056**	**.0320**	**.0615**	**.0715**	**.0789**	**.0759**	**.1005**	**.1279**
BCQ-Stats	.0146	.0523	.0896	.1491	.0146	.0272	.0321	.0362	.0334	.0454	.0603
CQL-Mean	.0284	.1085	.1809	.2845	.0284	.0551	.0646	.0717	.0682	.0915	.1175
CQL-Recency	.0243	.0908	.1536	.2412	.0243	.0464	.0546	.0606	.0573	.0774	.0995
CQL-Quality	.0244	.0928	.1563	.2524	.0244	.0472	.0555	.0621	.0584	.0788	.1030
CQL-Vanilla	.0118	.0478	.0838	.1409	.0118	.0237	.0284	.0323	.0296	.0412	.0555
Improvements over GRU4Rec (%)											
BC-Mean	+25.2	+2.5	−3.1	−7.2	+25.2	+10.2	+6.5	+4.2	+7.1	+2.0	−1.9
BC-Recency	+22.9	+0.2	−6.8	−10.2	+22.9	+8.0	+3.7	+1.5	+4.8	−1.2	−4.9
BC-Quality	+27.4	+2.8	−3.5	−7.2	+27.4	+11.3	+7.2	+4.7	+7.9	+2.1	−1.6
BC-Stats	−38.2	−48.1	−50.6	−50.8	−38.2	−45.0	−46.4	−47.0	−46.2	−48.4	−49.1
BCQ-Mean	**+43.0**	**+15.9**	**+6.7**	**+1.8**	**+43.0**	**+25.4**	**+20.0**	**+16.9**	**+21.5**	**+13.7**	**+8.8**
BCQ-Recency	**+49.3**	**+16.3**	**+7.0**	**+2.5**	**+49.3**	**+27.6**	**+21.8**	**+18.8**	**+23.1**	**+14.8**	**+10.1**
BCQ-Quality	**+43.5**	**+16.1**	**+6.8**	**+1.9**	**+43.5**	**+25.8**	**+20.2**	**+17.1**	**+21.8**	**+13.8**	**+9.0**
BCQ-Stats	−34.5	−49.4	−51.4	−50.3	−34.5	−44.4	−46.1	−46.3	−46.4	−48.6	−48.6
CQL-Mean	+27.2	+4.9	−1.9	−5.1	+27.2	+12.7	+8.6	+6.3	+9.5	+3.6	+0.2
CQL-Recency	+8.8	−12.2	−16.7	−19.5	+8.8	−5.1	−8.2	−10.1	−8.0	−12.3	−15.2
CQL-Quality	+9.4	−10.2	−15.2	−15.8	+9.4	−3.4	−6.7	−7.9	−6.2	−10.8	−12.2
CQL-Vanilla	−47.1	−53.8	−54.5	−53.0	−47.1	−51.5	−52.3	−52.1	−52.5	−53.4	−52.7

Table 5.1: Performance comparison between GRU4Rec baseline, BC, BCQ and CQL variants.

BCQ-Recency shows the best performance across most metrics (+49.3% Recall@1, +27.6% MRR@5, and +23.1% NDCG@5 compared to GRU4Rec). BCQ-Quality and BCQ-Mean follow with comparable improvements. BC-Quality shows moderate gains (27.4% Recall@1, 11.3% MRR@5, and 7.9% NDCG@5), while CQL-Mean achieves 27.2% Recall@1 and 12.7% MRR@5. Statistical variants perform worse than the other algorithms, with degradations ranging from -34.5% to -47.1% in Recall@1.

As shown in Figure 5.1, KG-enhanced variants demonstrate smooth convergence across all algorithms, whereas statistical variants exhibit poor convergence behavior. The differences in absolute loss magnitude reflect the optimization objectives of each algorithm: CQL maintains higher loss due to its conservative penalty; BCQ's loss reflects the joint learning of the behavioral cloning and Q-learning components; and BC's loss stems from the cross-entropy classification over the large action space.

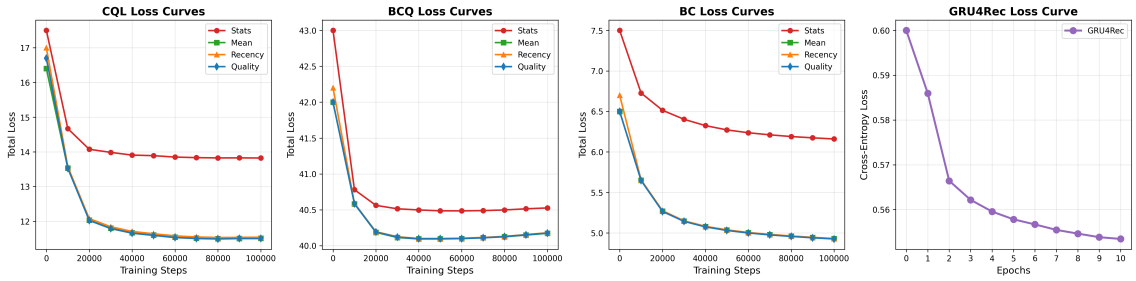

Figure 5.1: Training loss curves comparison across tested models.

5.2 Main Findings

RQ1: Integration Strategy Effectiveness Our investigation revealed that the effectiveness of knowledge graph integration strategies depends heavily on the algorithm used, and that distinct optimal approaches exist for each method.

BC achieves optimal performance with quality-based weighting (BC-Quality: +27.4% Recall@1, +11.3% MRR@5 over GRU4Rec). This aligns with BC's imitation learning objective. By emphasizing high-satisfaction interactions through quality weights, the model learns to distinguish strong positive signals from weaker ones in the demonstration data. The supervised learning nature of BC benefits directly from this explicit preference signal in the state representation.

With recency-based weighting, BCQ demonstrates superior performance (+49.3% Recall@1, +27.6% MRR@5). The temporal emphasis synergizes with the BCQ hybrid architecture. Recency weighting increases the likelihood that the action filtering mechanism retains contextually relevant items in the constrained action set. Then, the Q-learning component can optimize among these recent, relevant candidates. This results in the strongest overall performance across all tested configurations.

CQL performs best with mean aggregation (CQL-Mean: +27.2% Recall@1, +12.7% MRR@5). However, CQL's conservative penalty mechanism interacts poorly with biased state encodings.

Recency and quality weighting can inflate Q-value estimates for rarely seen state-action pairs, leading to excessive penalization. Mean aggregation produces smoother, more stable state representations that avoid triggering overly conservative value estimates.

Training dynamics confirmed these patterns. Knowledge graph variants demonstrated smooth convergence while maintaining stable loss trajectories across all algorithms, as shown in Figure 5.1. These results show that knowledge graph integration strategies should be tailored to the underlying learning mechanism. The interaction between state representation bias and algorithmic properties (e.g., imitation versus value-based learning, action constraints, and conservatism) fundamentally determines the effectiveness in ranking. Figure 5.2 illustrates the difference in ranking performance across all variants in each RL algorithm tested.

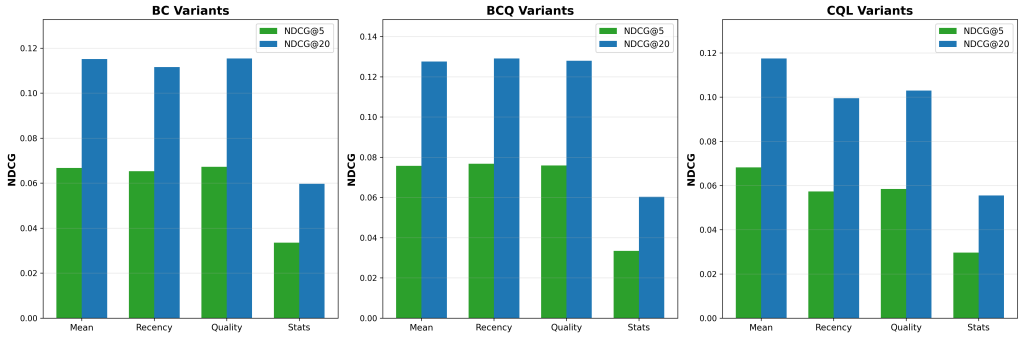

Figure 5.2: Performance comparison across aggregation strategies.

RQ2: Semantic versus Statistical Representations A comparison of statistical features and knowledge graph embeddings clearly shows that semantic representations are superior. Statistical variants perform poorly across all algorithms, showing severe degradation in performance. Compared to GRU4Rec, BC-Stats shows -38.2% Recall@1 and -45.0% MRR@5, BCQ-Stats shows -34.5% Recall@1 and -44.4% MRR@5, and CQL-Vanilla shows -47.1% Recall@1 and -51.5% MRR@5. Traditional aggregated statistics, such as interaction counts, rating averages, and popularity metrics, fundamentally lack the semantic relationship information necessary for effective sequential recommendation within RL frameworks.

The semantic versus statistical gap is particularly pronounced for value-based methods. BCQ variants with TransE embeddings improve upon the baseline by 43.0-49.3%, while BCQ-Stats decreases performance by 34.5%. This difference shows the importance of structured semantic representations for stable value function approximation in sparse recommendation environments. In BCQ/CQL, this estimator must generalize across state-action pairs, many of which are rarely observed in offline data. Statistical features only provide superficial similarity signals, while knowledge graph embeddings encode deep semantic relationships that enable meaningful generalization to unseen combinations.

Figure 5.3 shows the comparison of the best variant of each model, including the statistical model, in NDCG. The superior performance of the value-based algorithms and the poor performance of the statistical variant are evident.

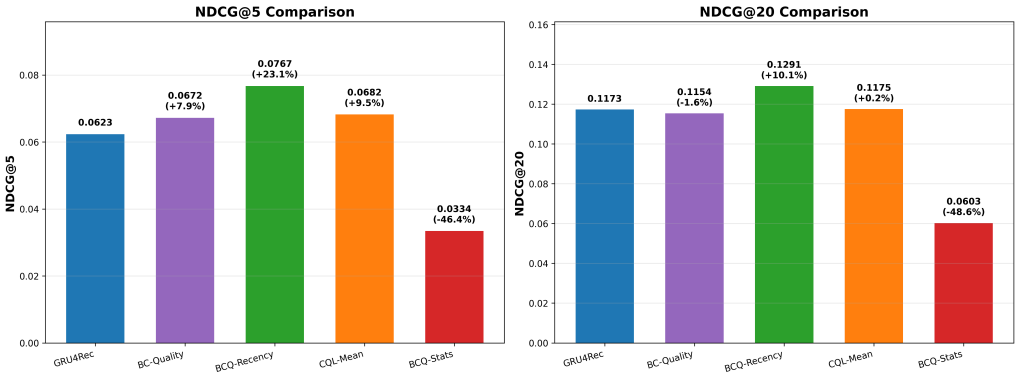

Figure 5.3: Performance comparison between knowledge graph-enhanced and statistical baseline variants.

5.3 Algorithm-Specific Analysis

Behavioral Cloning: Precision-Recall Trade-offs The BC variants demonstrate strong performance in terms of the top-rank metrics (Recall@1 and MRR@k), but their performance declines at higher k values. BC-Quality achieves a +27.4% increase in Recall@1, but a -7.2% decrease in Recall@20, compared to GRU4Rec. This pattern reveals a fundamental characteristic of the pointwise training approach used in this study.

BC is trained with pointwise supervision, learning to predict the single observed item at each stage. Although this enables BC to excel at identifying the most likely next item, it creates challenges when ranking larger sets of alternatives. This limitation comes from two factors: BC's cross-entropy objective optimizes predicting the demonstrated action rather than ranking multiple potentially relevant items, and the MovieLens dataset containing pointwise data rather than ranked lists, providing no supervision signal about the relevance of alternative items beyond the demonstrated one. The decreasing performance at higher k values reflects the mismatch between the pointwise training objective and the multi-item ranking evaluation task.

BCQ: Hybrid Architecture Advantages BCQ achieves the strongest overall performance, with BCQ-Recency reaching +49.3% Recall@1 and maintaining positive improvements even at Recall@20 (+2.5%). This success derives from BCQ's hybrid design that combines behavioral cloning and Q-learning.

The Q-learning component enables value-based optimization beyond directly observed behaviors, allowing BCQ to identify relevant items not explicitly demonstrated in similar contexts. The action filtering mechanism, which constrains the policy to select from actions similar to the behavior policy, prevents the value function from exploiting spurious high Q-values for rarely seen actions. This constrained exploration within the supported action space produces superior performance across all rank positions.

CQL: Conservative Penalties in Sparse Spaces CQL exhibits limited improvements, with only CQL-Mean providing consistent gains. Other CQL variants underperform at higher k values, with CQL-Recency showing -19.5% Recall@20.

The large action space (3,199 items) combined with sparse state-action coverage in offline data creates challenges for CQL's conservative mechanism. The penalty term, designed to prevent overestimation of out-of-distribution action values, becomes overly aggressive when most state-action pairs are rarely observed. Biased state representations (recency/quality weighting) exacerbate this issue by creating Q-value estimates that trigger excessive penalization.

Interestingly, stronger conservatism (higher α values) improved CQL performance in preliminary experiments, suggesting that the sparse recommendation setting requires more aggressive constraints to prevent value overestimation for pairs rarely seen. However, this comes at the cost of potentially undervaluing legitimately relevant but infrequent recommendations.

GRU4Rec: Sequential Pattern Learning GRU4Rec demonstrates competitive performance in broader retrieval tasks, achieving the highest Recall@20 among all methods tested. Its recurrent architecture's ability to capture sequential co-occurrence patterns across user sessions gives GRU4Rec an advantage, making it particularly effective when retrieving larger candidate sets.

However, GRU4Rec's overall metrics lag behind those of KG-enhanced RL methods, especially in terms of top-rank precision. For example, GRU4Rec's Recall@1 and MRR@5, are substantially lower than BCQ-Recency's. The performance gap narrows at higher k values, where GRU4Rec's Recall@20 surpasses several knowledge graph variants, including BC-Quality and BC-Recency.

This divergent performance across ranking positions reflects a fundamental trade-off: GRU4Rec excels at learning which items follow others in sessions, but it lacks explicit semantic understanding of why these transitions occur. KG embeddings encode relationships, such as shared genres, directors, or themes, that enable more precise top-k recommendations through semantic generalization beyond observed co-occurrence patterns.

Additionally, GRU4Rec offers practical advantages in deployment scenarios. Its training time (16 minutes for 10 epochs) is significantly lower than that of offline RL methods (90–160 minutes for 100,000 steps) plus the time taken to train TransE, and its simpler architecture requires less hyperparameter tuning and is easier to maintain in production systems. These efficiency benefits may outweigh the disadvantage of lower precision in applications where broader retrieval or computational constraints are priorities.

The performance difference may also reflect the data scale requirements. Recurrent architectures generally require larger datasets to learn robust sequential patterns, and the relatively sparse MovieLens dataset may not provide enough information for GRU4Rec to demonstrate its full capabilities. Future work could investigate whether improving GRU4Rec's performance requires additional training data, longer session histories, or architectural enhancements.

Chapter 6

Conclusion and Future Work

This experimental study examined the intersection of three fundamental areas of artificial intelligence: Recommender Systems, RL, and Representation Learning. A comprehensive literature review of these three domains was conducted at the beginning of the research, which led to the study of KG-enhanced offline RL for sequential recommendation tasks.

The investigation then proceeded toward next-item prediction using the MovieLens dataset. KGs were obtained from open-source communities, such as DBpedia, and encoded into dense vector representations using the TransE algorithm. The core focus of the experiments was on how different aggregation functions for combining KG embeddings into state representations affect the performance of three offline RL algorithms: Behavioral Cloning (BC), Batch Constrained Q-learning (BCQ), and Conservative Q-learning (CQL).

As baseline for comparison, we employed GRU4Rec, an established recurrent neural network approach for sequential recommendation. This approach was used alongside a statistical state representations that aggregate traditional features without KG enhancement. This dual baseline approach allowed us to evaluate both algorithmic advances between RNN-based and RL algorithms and representational improvements with the KGs.

Our investigation revealed that the optimal KG integration strategy depends on the algorithm. BC performed best with quality-weighted embeddings (+27.4% Recall@1 over GRU4Rec), BCQ excelled with recency-weighted representations (+49.3% Recall@1), and CQL achieved optimal results with mean aggregation (+27.2% Recall@1). These results reflect the interaction between state representation bias and algorithmic properties: imitation learning benefits from explicit preference signals, hybrid architectures leverage temporal patterns, and conservative methods require balanced representations to avoid excessive penalization.

A comparison of semantic and statistical representations clearly demonstrated the superiority of KG embeddings. Semantic variants improved upon the baseline by 27-49%, while statistical variants decreased Recall@1 performance by 34-47%. This difference was most apparent in value-based methods: BCQ with TransE embeddings improved by 43-49%, whereas BCQ-Stats decreased by 34.5%. Training loss confirmed that traditional aggregated statistics lack the semantic relationship information necessary for effectively approximating the value function in sparse

recommendation environments.

Overall, BCQ was the most effective approach, as its hybrid architecture successfully combined BC's semantic understanding with Q-learning's exploration. BC variants excelled at top-rank precision but showed declining performance at higher cutoffs. This reflects the mismatch between BC's pointwise training objective and the ranking evaluation task. CQL faced challenges in sparse, large-action-space settings; only CQL-Mean provided consistent improvements. These findings demonstrate that successful offline RL for recommendations requires semantic state representations and careful alignment between integration strategies and algorithmic learning.

6.1 Study Limitations

Several limitations constrain the scope and generalizability of this study.

Limited Knowledge Graph Representation Models The study focused exclusively on TransE embeddings for KG representation without exploring alternative relational models, such as TransR, RotatE, or graph neural network approaches, which might capture different semantic relationships more effectively.

Embedding Quality Assessment This study trained KG embeddings to evaluate how RL methods behave when the state contained more dense representations. However, it did not fully evaluated the embeddings quality with techniques such as link prediction accuracy or semantic cluster validation.

Evaluation Metric Limitations The evaluation framework relied solely on traditional ranking metrics (Recall@k, MRR@k, and NDCG@k), which may not adequately capture the semantic contributions of KG embeddings. These positional accuracy measures cannot assess whether recommendations demonstrate semantic coherence, diversity, or meaningful relationships to user preferences beyond ranking position. This limitation constrains the understanding of how different representation models contribute to recommendation quality, establishing the need for semantic-aware evaluation methodologies.

Dataset and Coverage Limitations The evaluation was restricted to a single dataset (MovieLens-1M), which limits the generalizability of the findings to different domains and user behavior patterns. Additionally, items absent from the KG were excluded from the study, resulting in a reduction in data that may not reflect real-world scenarios where partial knowledge coverage is common.

Limited Offline RL Algorithm Coverage The offline RL investigation was limited to three algorithms: BC, BCQ, and CQL. These algorithms represent only a subset of the available approaches. Other more recent offline RL algorithms may demonstrate different performance characteristics when integrated with KGs. Additionally, modern recommendation systems increasingly rely on user simulators for policy evaluation and training, a factor that was not considered in this study. Without simulated environments, it is difficult to conduct controlled experiments, evaluate counterfactual scenarios, or assess the long-term impact of different recommendation policies on user engagement and satisfaction.

Simple Aggregation Strategies The study used relatively simple aggregation strategies to combine historical embeddings. More sophisticated approaches, such as attention-based mechanisms, could provide different insights into optimal KG integration.

Missing KERL Comparison This study did not include a direct comparison with KERL Wang et al. (2020), cited in Section 2.4. While KERL employs a different approach, such a comparison would have provided valuable insights into the effectiveness of both reward modeling and knowledge-based future preference detection. Future work should incorporate KERL and similar hybrid approaches to establish a more comprehensive baseline comparison.

6.2 Future Work

6.2.1 Near-Term Research Directions

Knowledge Graph Embedding Exploration: To identify which relational models best capture semantic relationships for sequential recommendation tasks in next-item prediction, a comprehensive ablation study should investigate different KG embedding methods beyond TransE, including TransR, RotatE, ComplEx, and DistMult. The addition of negative edges (e.g. $(u_i, \text{disliked}, m_j)$) in the KG to explicitly show negative feedback is also a possibility.

Advanced Aggregation Mechanisms: Future work should explore more sophisticated methods of combining historical embeddings. One promising approach is attention-based aggregation, which can dynamically adjust the importance of different historical interactions based on context and item relationships.

Expanded Evaluation: Evaluating these approaches across a larger set of dataset variants and diverse domains (e.g., e-commerce, music streaming, and video recommendations) would demonstrate their scalability and effectiveness in specific domains.

Semantic-Aware Evaluation Metrics: The development of recommendation quality metrics designed specifically to evaluate the effectiveness of different representation models is critically needed. These metrics could assess the semantic coherence of recommendation lists, measure the semantic similarity between recommended and ground-truth items, and evaluate how well different embedding approaches capture user preference patterns beyond positional accuracy.

6.2.2 Long-Term Research Directions

User Modeling and Simulation: Integration with sophisticated user simulators such as DeepFM Guo et al. (2017) would enable more controlled experimentation and policy evaluation. These simulators could provide richer feedback signals and facilitate counterfactual analyses of recommendation strategies. More importantly, they could address the state-action sparsity problem by generating synthetic interactions for underrepresented state-action combinations. This would enable the systematic exploration of the action space and the more robust training of value-based RL algorithms.

Sample-Efficient RL Algorithms: Investigating newer, more sample-efficient offline RL algorithms, such as model-based approaches like MOPO Yu et al. (2020) and COMBO Yu et al. (2021), or recent advances in conservative learning, could potentially reduce computational overhead while maintaining or improving performance.

Path Reasoning and Explainable Recommendations: Investigating RL-based path reasoning methods that walk through KG structures to generate recommendations. These methods could generate explainable recommendations by revealing the paths taken through the KG. This allows users to understand why specific items were recommended based on the semantic connections discovered by the model.

Reward Function Modeling: Lastly, developing multi-objective reward functions that incorporate KG coherence terms alongside traditional user satisfaction metrics. An approach could be to penalize recommendations that violate semantic relationships and reward recommendation sequences that maintain coherent paths through the KG structure. These approaches would allow RL algorithms to optimize for both immediate user engagement and long-term semantic consistency of the recommendation policy.

References

Lada A Adamic and Eytan Adar. Friends and neighbors on the web. In *Social networks*, volume 25, pages 211–230, 2003.

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

M. Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *ACM Comput. Surv.*, 55(7), December 2022. ISSN 0360-0300. doi: 10.1145/3543846. URL https://doi.org/10.1145/3543846.

Mohammad M. Afsar et al. Addressing challenges in reinforcement learning for recommender systems with conservative objectives. *arXiv preprint arXiv:2305.18820*, 2023.

Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. Pykeen 1.0: A python library for training and evaluating knowledge graph embeddings. *Journal of Machine Learning Research*, 22(82):1–6, 2021. URL http://jmlr.org/papers/v22/20-825.html.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, page 722–735, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3540762973.

Giacomo Balloccu, Ludovico Boratto, Gianni Fenu, and Mirko Marras. Reinforcement recommendation reasoning through knowledge graphs for explanation path quality. *Knowledge-Based Systems*, 258:109947, 2022.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286 (5439):509–512, 1999.

Oren Barkan and Noam Koenigstein. Item2vec: Neural item embedding for collaborative filtering. *CoRR*, abs/1603.04259, 2016. URL http://arxiv.org/abs/1603.04259.

Payam Barnaghi, Wei Wang, Cory Henson, and Kerry Taylor. Semantics for the internet of things: early progress and back to the future. *International Journal on Semantic Web and Information Systems*, 8(1):1–21, 2012.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795, Red Hook, NY, USA, 2013. Curran Associates Inc.

Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

Pedro G Campos, Fernando Díez, and Iván Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1-2):67–119, 2014.

Bohan Chen and Andrea L. Bertozzi. Autokg: Efficient automated knowledge graph generation for language models, 2023. URL https://arxiv.org/abs/2311.14740.

Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. Large-scale interactive recommendation with tree-structured policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3312–3320, 2019a.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. Top-k off-policy correction for a reinforce recommender system. pages 456–464, 2019b.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 456–464. ACM, 2019c.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607, 2020.

Xiangyu Chen, Hao Ma, Ji Wan, Bo Li, and Ting Xia. A survey on reinforcement learning for recommender systems. *arXiv preprint arXiv:2109.10665*, 2021a.

Xiaocong Chen, Lina Yao, Julian J. McAuley, Guanglin Zhou, and Xianzhi Wang. A survey of deep reinforcement learning in recommender systems: A systematic review and future directions. *CoRR*, abs/2109.03540, 2021b. URL https://arxiv.org/abs/2109.03540.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM, 2016. doi: 10.1145/2988450.2988454. URL https://dl.acm.org/doi/10.1145/2988450.2988454.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. ACL, 2014a. URL https://arxiv.org/abs/1406.1078.

Kyunghyun Cho, Bart Van Merriënboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014b. doi: 10.3115/v1/D14-1179.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*, pages 87–94, 2008.

Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, and Yong Li. Graph neural networks for recommender systems: Challenges, methods, and directions. *CoRR*, abs/2109.12843, 2021. URL https://arxiv.org/abs/2109.12843.

David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016. URL http://arxiv.org/abs/1607.00653.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for CTR prediction. *CoRR*, abs/1703.04247, 2017. URL http://arxiv.org/abs/1703.04247.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568, 2020.

William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

Ruining He and Julian J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *CoRR*, abs/1602.01585, 2016. URL http://arxiv.org/abs/1602.01585.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004. ISSN 1046-8188. doi: 10.1145/963770.963772. URL https://doi.org/10.1145/963770.963772.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th international conference on learning representations*, 2015.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016a. URL https://arxiv.org/abs/1511.06939.

Balazs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016b.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys*, 54(4):1–37, 2021.

Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S. Yu. Leveraging meta-path based context for top- n recommendation with a neural co-attention model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 1531–1540, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3219965. URL https://doi.org/10.1145/3219819.3219965.

Eugene Ie, Chih wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. Recsim: A configurable simulation platform for recommender systems. 2019.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. On offline evaluation of recommender systems. *CoRR*, abs/2010.11060, 2020. URL https://arxiv.org/abs/2010.11060.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE international conference on data mining*, pages 197–206, 2018.

Thomas N. Kipf and Max Welling. Variational graph auto-encoders, 2016. URL https://arxiv.org/abs/1611.07308.

Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53 (4):89–97, 2010.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Karl Krauth, Sarah Dean, Alex Zhao, Wenshuo Guo, Mihaela Curmei, Benjamin Recht, and Michael I. Jordan. Do offline metrics predict online performance in recommender systems? *CoRR*, abs/2011.07931, 2020. URL https://arxiv.org/abs/2011.07931.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. In *International journal of computer vision*, volume 123, pages 32–73, 2017.

Paul Krzakala, Gabriel Melo, Charlotte Laclau, Florence d'Alché Buc, and Rémi Flamary. The quest for the graph level autoencoder (grale), 2025. URL https://arxiv.org/abs/2505.22109.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191, 2020.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, 2016.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, page 556–559, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581137230. doi: 10.1145/956863.956972. URL https://doi.org/10.1145/956863.956972.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Feb. 2015. doi: 10.1609/aaai.v29i1.9491. URL https://ojs.aaai.org/index.php/AAAI/article/view/9491.

Yuanguo Lin, Yong Liu, Fan Lin, Lixin Zou, Pengcheng Wu, Wenhua Zeng, Huanhuan Chen, and Chunyan Miao. A survey on reinforcement learning for recommender systems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(10):13164–13184, October 2024. ISSN 2162-2388. doi: 10.1109/tnnls.2023.3280161. URL http://dx.doi.org/10.1109/TNNLS.2023.3280161.

Shijie Liu, Nan Zheng, Hui Kang, Xavier Simmons, Junjie Zhang, Matthias Langer, Wenjing Zhu, Minseok Lee, and Zehuan Wang. Embedding optimization for training large-scale deep learning recommendation systems with embark. In *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys '24, page 622–632, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705052. doi: 10.1145/3640457.3688111. URL https://doi.org/10.1145/3640457.3688111.

Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

Yaxiong Liu, Jiecao Liang, Jianqiang Liu, and Ke Li. Reinforcement learning over sentiment-augmented knowledge graphs towards accurate and explainable recommendation. pages 688–696, 2020.

Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

Malte Ludewig and Dietmar Jannach. Evaluation of session-based recommendation algorithms. In *User modeling and user-adapted interaction*, volume 28, pages 331–390, 2018.

Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM computing surveys*, 49(4):1–33, 2016.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013a. URL http://arxiv.org/abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. URL https://api.semanticscholar.org/CorpusID:205242740.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. pages 1928–1937, 2016.

Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204, 2000.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

Liwei Pan, Weike Pan, Meiyan Wei, Hongzhi Yin, and Zhong Ming. A survey on sequential recommendation, 2025. URL https://arxiv.org/abs/2412.12770.

Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. In *Artificial Intelligence Review*, volume 13, pages 393–408. Springer, 1999.

Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652, 2014. URL http://arxiv.org/abs/1403.6652.

Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

Sunita Barve Poonam B. Thorat, R. M. Goudar. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4):31–36, January 2015. ISSN 0975-8887. doi: 10.5120/19308-0760. URL https://ijcaonline.org/archives/volume110/number4/19308-0760/.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461, 2009.

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 811–820, New York, NY, USA, 2010. ACM. doi: 10.1145/1772690.1772773.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. *CoRR*, abs/1205.2618, 2012. URL http://arxiv.org/abs/1205.2618.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Amany Sami, Waleed El Adrousy, Shahenda Sarhan, and Samir Elmougy. A deep learning based hybrid recommendation model for internet users. *Scientific Reports*, 14(1):29390, 2024. doi: 10.1038/s41598-024-79011-z. URL https://www.nature.com/articles/s41598-024-79011-z.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

Markus Schedl. The lfm-1b dataset for music retrieval and recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, ICMR '16, page 103–110, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450343596. doi: 10.1145/2911996.2912004. URL https://doi.org/10.1145/2911996.2912004.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Nikhil Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *international conference on machine learning*, pages 1670–1679, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Juan F Sequeda and Daniel P Miranker. Ultrawrap: Sparql execution on relational data. In *Journal of Web Semantics*, volume 22, pages 19–39, 2013.

Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011. ISBN 978-0-387-85819-7. doi: 10.1007/978-0-387-85820-3_8. URL http://dx.doi.org/10.1007/978-0-387-85820-3_8.

Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 269–272, 2010.

Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009, January 2009. ISSN 1687-7470. doi: 10.1155/2009/421425. URL https://doi.org/10.1155/2009/421425.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2nd edition, 2018.

Richard Stuart Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, 1984. AAI8410337.

Xuehan Tang, Zhao Du, Xiangnan He, Fajie Yuan, Qi Tian, and Tat-Seng Chua. Hierarchical fashion graph network for personalized outfit recommendation. pages 159–168, 2019.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, volume 48, pages 2071–2080, 2016.

Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion, 2017. URL https://arxiv.org/abs/1706.02263.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015. URL http://arxiv.org/abs/1509.06461.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

Ellen M Voorhees. The trec-8 question answering track report. *TREC*, 99:77–82, 1999.

Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, September 2014. ISSN 0001-0782. doi: 10.1145/2629489. URL http://doi.acm.org/10.1145/2629489.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Ripple network: Propagating user preferences on the knowledge graph for recommender systems. *CoRR*, abs/1803.03467, 2018. URL http://arxiv.org/abs/1803.03467.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. pages 3307–3313, 2019a.

Kai Wang, Zhene Zou, Yue Shang, Qilin Deng, Minghao Zhao, Runze Wu, Xudong Shen, Tangjie Lyu, and Changjie Fan. Rl4rs: A real-world benchmark for reinforcement learning based recommender system. *ArXiv*, abs/2110.11073, 2021.

Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, Shaozhang Niu, and Jimmy Huang. Kerl: A knowledge-guided reinforcement learning model for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 209–218, 2020.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12): 2724–2743, 2017.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. KGAT: knowledge graph attention network for recommendation. *CoRR*, abs/1905.07854, 2019b. URL http://arxiv.org/abs/1905.07854.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174, 2019c.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Shiwen Wu, Wentao Zhang, Fei Sun, and Bin Cui. Graph neural networks in recommender systems: A survey. *CoRR*, abs/2011.02260, 2020. URL https://arxiv.org/abs/2011.02260.

Shiwen Wu, Fei Sun, Wentao Zhang, Xing Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

Zhengzhong Wu and Martha Palmer. Open information extraction on scientific text: An evaluation. *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3049–3060, 2018.

Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. Reinforcement knowledge graph reasoning for explainable recommendation. *CoRR*, abs/1906.05237, 2019a. URL http://arxiv.org/abs/1906.05237.

Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. Reinforcement knowledge graph reasoning for explainable recommendation. pages 285–294, 2019b.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases, 2015. URL https://arxiv.org/abs/1412.6575.

Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. Exploring large language models for knowledge graph completion, 2024. URL https://arxiv.org/abs/2308.13916.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: model-based offline policy optimization. *CoRR*, abs/2005.13239, 2020. URL https://arxiv.org/abs/2005.13239.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: conservative offline model-based policy optimization. *CoRR*, abs/2102.08363, 2021. URL https://arxiv.org/abs/2102.08363.

Yuanqing Yu, Chongming Gao, Jiawei Chen, Heng Tang, Yuefeng Sun, Qian Chen, Weizhi Ma, and Min Zhang. Easyrl4rec: A user-friendly code library for reinforcement learning based recommender systems, 2024.

Eva Zangerle and Christine Bauer. Evaluating recommender systems: Survey and framework. *ACM Comput. Surv.*, 55(8), December 2022. ISSN 0360-0300. doi: 10.1145/3556536. URL https://doi.org/10.1145/3556536.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, 2014.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16,

page 353–362, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939673. URL https://doi.org/10.1145/2939672.2939673.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31:5165–5175, 2018.

S. Zhang, L. Chen, C. Wang, S. Li, and H. Xiong. Temporal graph contrastive learning for sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9359–9367, 2024.

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52(1), February 2019a. ISSN 0360-0300. doi: 10.1145/3285029. URL https://doi.org/10.1145/3285029.

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys*, 52(1):1–38, 2019b.

Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Liqiang Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. pages 1040–1048, 2018.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. pages 167–176, 2018.

Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1059–1068, 2018.

Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, volume 33, pages 5941–5948, 2019. doi: 10.1609/aaai.v33i01.33015941. URL https://ojs.aaai.org/index.php/AAAI/article/view/4545.

Sijin Zhou, Xinyi Dai, Haokun Chen, Weinan Zhang, Kan Ren, Ruiming Tang, Xiuqiang He, and Yong Yu. Interactive recommender system via knowledge graph-enhanced reinforcement learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 179–188, 2020.

Tao Zhou, Jie Ren, Matúš Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. In *Physical Review E*, volume 76, page 046115. APS, 2007.