Path-Level Explainability in Knowledge Graph Recommenders

Decoding Recommendations: Insights from Knowledge Graph Paths

Shuang Liu

Delft University of Technology

TU Delft

Path-Level Explainability in Knowledge Graph Recommenders

Decoding Recommendations: Insights from Knowledge Graph Paths

by

Shuang Liu

Student Name: Shuang Liu

Student Number: 5053005

Thesis advisor: Megha Khosla
Daily supervisor: Masoud Mansoury
Faculty: Electrical Engineering, Mathematics and Computer Science, Delft

TU Delft

Abstract

Knowledge Graph-based Recommender Systems (KGRS) have attracted significant attention due to their potential to enhance both recommendation accuracy and interpretability by leveraging structured relational knowledge. Despite the widespread use of reasoning paths as explanations, systematic evaluation of explanation quality across different KGRS paradigms, and its relationship with recommendation correctness, remains limited. This study provides a comprehensive empirical analysis of three representative KGRS paradigms, including path-based, embedding-based, and hybrid methods, focusing on multiple dimensions of explanation quality, including temporal relevance, popularity, diversity, semantic consistency, and faithfulness.

The analysis investigates differences in explanation characteristics across paradigms, revealing that models such as TMER achieve high recommendation accuracy through constrained path structures, whereas reinforcement learning-based models, including TPRec and PGPR, generate explanations with stronger lexical alignment to user review rationales. The study also examines explanation quality for correctly recommended (relevant) and incorrectly recommended (irrelevant) items. The results show that explanation-ground truth consistency metrics (Precision, Recall, and F1) exhibit greater disparities between correctly and incorrectly recommended items than other evaluation metrics. In RippleNet, the impact of ripple set size and explanation-oriented neighbor sampling strategies on recommendation performance and explanation quality is analyzed. Non-uniform sampling guided by temporal relevance, popularity, or diversity effectively shapes ripple sets to enhance explanation properties without significantly affecting overall recommendation accuracy.

The findings highlight trade-offs between recommendation accuracy and explanation quality, demonstrating that careful model design and sampling strategies can produce interpretable, user-aligned recommendations while maintaining high performance. These insights provide guidance for developing KGRS capable of delivering accurate predictions accompanied by semantically rich, temporally relevant, and user-preferred explanations, thereby improving transparency, trust, and user satisfaction in real-world applications.

Contents

1

Introduction

1.1. Context

In today's digital era, users are constantly exposed to an overwhelming amount of online information, ranging from movies and music to e-commerce products and social media content. To mitigate this information overload, recommender systems have become essential tools that suggest personalized items based on user preferences and behavioral patterns. Traditional systems, however, often operate as black boxes, focusing primarily on predictive accuracy without providing insights into their internal decision-making processes [41]. This lack of transparency can reduce user trust and limit interpretability.

Among various recommender paradigms, *Knowledge Graph-based Recommender Systems* (KGRS) have emerged as a promising approach. A Knowledge Graph (KG) is a heterogeneous graph where nodes represent entities and edges encode semantic relations [2]. KGRS leverage these structured relationships to enrich user and item representations and enable multi-hop reasoning, which can improve both recommendation accuracy and interpretability [31].

Research on KGRS has evolved through several methodological paradigms, reflecting the dual goals of improving recommendation performance and enhancing interpretability. **Embedding-based approaches** are among the earliest and most widely explored methods. They learn continuous vector representations of entities and relations in the knowledge graph using knowledge graph embedding (KGE) techniques, which are then incorporated into recommendation models to enrich user and item features [40]. These methods are effective at alleviating challenges such as sparse interaction data and cold-start problems, and they can capture latent semantic connections across the graph.

Path-based approaches explicitly model reasoning paths in the knowledge graph to connect users and items. These methods often mine or pre-define relational sequences, which are useful for linking user preferences with recommendations [29]. By following these paths, the models can provide naturally interpretable explanations, showing not just what is recommended but also why, based on the structure of the knowledge graph.

To leverage the strengths of both paradigms, **hybrid methods** have been proposed. These models integrate both latent embeddings learned from the knowledge graph and explicit multi-hop connectivity information, aiming to capture both semantic relevance and structural relationships. However, hybrid methods can be computationally more expensive due to multi-hop propagation and embedding updates[13].

Many models following these paradigms generate reasoning paths for each recommendation. In KGRS, such reasoning paths not only improve recommendation accuracy by capturing higher-order interactions, but also provide the foundation for explainability. For example, a recommended item can be justified through a reasoning path such as:

$$\text{User}_{u_1} \xrightarrow{\textit{purchased}} \text{Product}_{p_1} \xrightarrow{\textit{belongs\_to}} \text{Category}_{c_1} \xleftarrow{\textit{belongs\_to}} \text{Product}_{p_2}$$

which explicitly shows how a recommended item is connected to a user's past behavior, enhancing transparency and trust [34].

Recent studies further propose multi-dimensional metrics to assess explanation quality, with one commonly used dimension capturing temporal relevance, popularity, and diversity [1]. Specifically, temporal relevance evaluates whether explanations reflect recent user interactions, popularity measures the extent to which explanations rely on widely known versus niche entities, and diversity assesses the variety of reasoning paths or explanation types provided to the user. These metrics offer a more nuanced perspective on interpretability.

1.2. Research Gap and Motivation

Despite advantages of KGRS mentioned before, several critical challenges remain. First, although KGRS can generate path-based explanations, their effectiveness across multiple dimensions, such as temporal relevance, popularity, diversity [1], semantic consistency [42], and faithfulness [43], has not been systematically evaluated. Comparisons of explanation quality across different KGRS paradigms, including path-based, reinforcement learning-based, and embedding-based approaches, are also limited, making it difficult to identify the strengths and weaknesses of each method.

Second, existing evaluation protocols typically aggregate results over the Top-K recommendation list without distinguishing between correctly recommended items (relevant items) and incorrectly recommended items (irrelevant items). Such aggregation can obscure important behaviors of explanation mechanisms and hinder the ability to assess whether explanations truly reflect the reasoning process or correlate with successful recommendations.

Third, there is limited research on leveraging explanation metrics to indicate recommendation correctness and guide adaptive recommendation strategies according to user preferences (e.g., favoring popular items during shopping). Integrating such adaptive mechanisms has the potential to generate more accurate, personalized explanations while simultaneously improving overall recommendation performance.

Addressing these gaps is crucial for several reasons. Systematically evaluating explanation quality across multiple dimensions and recommendation outcomes can provide a more reliable understanding of how different KGRS paradigms generate interpretable recommendations, guiding both model development and selection. Furthermore, understanding how explanation metrics relate to recommendation correctness enables adaptive strategies that align with user preferences, enhancing personalization and user satisfaction. Ultimately, this research contributes to the development of recommender systems that are not only accurate but also transparent, trustworthy, and capable of delivering meaningful explanations, thereby supporting more informed decision-making and improving user engagement in real-world applications.

1.3. Research Questions

Guided by the research gaps, this study addresses the following research questions:

- **RQ1:** *How do different KGRS model paradigms (path-based, embedding-based, and hybrid methods) differ in terms of explanation characteristics?*

  This question investigates how different KGRS paradigms generate explanations. Representative models from each paradigm are selected and evaluated using unified explainability metrics. By comparing their explanation characteristics, such as reasoning structure and diversity, this analysis aims to identify the strengths and limitations of each approach.

- **RQ2:** *How do explanation quality metrics differ between the explanations generated for correctly recommended (relevant) items and incorrectly recommended (irrelevant) items, and whether the adopted metrics can effectively distinguish the explanations of relevant items from those of irrelevant items?*

  The Top-K recommendation results are divided into relevant and irrelevant items, with their corresponding reasoning explanations evaluated separately. By comparing the metric distributions

between these two groups, this study investigates whether the adopted evaluation metrics can effectively differentiate explanations for correct recommendations from those for incorrect ones.

- **RQ3:** *How do the item neighborhood sample sizes and explanation-oriented neighborhood sampling strategies influence recommendation performance and the properties of generated explanations?*

  This research question explores the impact of explanation-aware design configurations on neighborhood sampling in graph-based recommendation models. Experiments are first conducted to analyze how different item neighborhood sample sizes affect overall recommendation performance. After selecting a suitable neighborhood scale, multiple explanation-metric guided neighbor sampling strategies are adopted. The subsequent evaluation examines their joint effects on recommendation accuracy and the inherent characteristics of generated reasoning explanations.

1.4. Thesis Structure

This thesis is organized as follows: Chapter 1 introduces the research background, problem context, and research questions. Chapter 2 reviews related work on knowledge graph-based recommender systems and explainability methods. Chapter 3 presents the methodology, including problem definition, datasets, evaluation metrics, and models. Chapter 4 reports experimental results, analyzing both recommendation performance and explanation quality, as well as their trade-offs and the impact of sampling strategies. Finally, the thesis concludes with key findings and future directions in Chapter 5.

2

Related Work

2.1. Learning paradigms in KGRS

Knowledge-graph based recommender systems (KGRS) have branched into several major learning paradigms. These paradigms capture different philosophies for representing graph structure, propagating signals, performing reasoning, and producing explanations. Below we describe three widely adopted paradigms—embedding-based, path-based, and hybrid—and discuss representative techniques, their strengths and weaknesses, and how each paradigm affects explainability in practice.

2.1.1. Embedding-based methods

Embedding-based methods learn continuous vector representations for entities and relations, capturing both local and high-order graph semantics within a low-dimensional latent space. These models propagate user and item signals through neighborhood aggregation, allowing the system to infer semantic similarity and structural connectivity indirectly. Graph neural networks (GNNs) and attention mechanisms are widely adopted to enhance this propagation process. For instance, the Knowledge Graph Attention Network (KGAT) introduces attention weighting to model the varying importance of neighboring entities and relations, which strengthens the expressiveness of node representations and provides interpretable attention weights that can serve as a form of local explanation [31]. Time-aware embedding models such as TPRec further integrate temporal dimensions into the embedding learning process, ensuring that user and item representations dynamically reflect recent interactions and contextual relevance [42].

These embedding-based paradigms are effective in achieving high recommendation accuracy and are typically trained end-to-end with ranking losses such as Bayesian Personalized Ranking (BPR) [21] or cross-entropy. However, despite their empirical success, their interpretability remains limited: attention scores or feature importance can offer approximate insight into influential relations, but these signals often lack faithfulness, meaning they do not necessarily correspond to the model's true reasoning process. Furthermore, because the embeddings exist in an abstract vector space, the resulting explanations are inherently latent and do not form explicit reasoning paths understandable to end users.

2.1.2. Path-based methods

Path-based methods aim to generate explicit multi-hop reasoning paths that connect users to recommended items within a knowledge graph, providing interpretable evidence for recommendations. Unlike embedding-based methods that compress the graph into a latent vector space, path-based approaches directly operate on the graph's structural topology. Typically, these approaches first extract user-item paths and then feed them into a recommendation learning framework to model connectivity patterns between users and items. Depending on the path extraction strategy, path-based methods can be divided into two main categories: (1) random-walk [9] based models, which sample or traverse paths probabilistically before using them for recommendation learning, and (2) meta-path based models, which leverage predefined meta-paths [26] to select meaningful paths. In particular, the TMER model explic-

itly defines both *user-item* and *item-item* meta-paths to capture temporal dependencies in sequential recommendations. TMER treats a user's historical purchased items and their inter-item transitions not simply as independent events but as interconnected paths in the knowledge graph: by modelling item-item paths between consecutive items, TMER enhances explanation power by capturing how one item leads to another, beyond the usual user-to-item connections [3].

Although path-based methods provide high interpretability and traceability, they face several challenges. The path search space in large graphs grows exponentially, requiring pruning, sampling, or learned navigation strategies that may introduce biases. Additionally, not all discovered paths faithfully reflect the model's decision-making process, raising concerns about explanation fidelity. Nonetheless, path-based reasoning remains one of the most transparent approaches for linking predictive performance with explainable semantics.

2.1.3. Hybrid methods

Between these two extremes lie hybrid methods, which combine the representation power of embedding-based models with the interpretability of path-based reasoning. Hybrid KGRS leverages embeddings for local and global graph structure, and incorporates extracted paths for richer context and higher interpretability [13]. These models often employ embeddings to guide path sampling or scoring, while the discovered paths provide semantic grounding for explanation. For example, the CAFE framework adopts a coarse-to-fine neural symbolic reasoning paradigm: it first generates a user profile as a coarse behavioral sketch and then guides a path-finding process that derives reasoning paths over a knowledge graph [36].

Hybrid methods, which combine the expressive power of embedding-based representation learning with the interpretability of explicit path-based reasoning, hold great potential for knowledge-graph-based recommender systems. In practical deployments, they offer enhanced flexibility, as embeddings can efficiently capture global and local graph structures while explicit paths provide interpretable evidence for recommendations. This combination allows hybrid models to leverage the strengths of both paradigms, balancing predictive accuracy with explainability. However, these architectures also introduce non-trivial challenges. Integrating multiple modules increases overall model and system complexity, making training more demanding, hyperparameter tuning more resource-intensive, and deployment less lightweight. Achieving an optimal balance between accurate predictions and rich interpretability remains a key challenge for hybrid KGRS.

2.2. Methodological tools used in KGRS

Knowledge-graph-based recommender systems (KGRS) integrate diverse methodological tools to enable reasoning, explainability, and dynamic adaptation. Beyond core learning paradigms such as embedding- or path-based methods, KGRS employ a set of complementary methodologies that provide control, structure, and interpretability. This section highlights four such methodological directions that have gained prominence in recent research: reinforcement learning for path search, rule learning and neural-symbolic reasoning, counterfactual and causal reasoning, and knowledge graph message passing.

2.2.1. Reinforcement Learning (RL) for path search

Reinforcement Learning (RL) has been increasingly employed to tackle the combinatorial nature of reasoning path search in knowledge graphs. In the context of recommendation, RL agents are trained to navigate the knowledge graph from a user node to a target item node through a sequence of semantically meaningful relations. Each navigation episode corresponds to a reasoning path, which can then serve as both a predictive and interpretable trace for recommendation.

A representative model, DeepPath [38], formulates path finding as a Markov decision process (MDP), where the agent incrementally selects relations and entities to maximize a reward associated with reaching correct or relevant targets. Similarly, MINERVA [4] extends this idea by employing policy gradients to learn reasoning trajectories conditioned on query entities and relations. In recommendation settings, RL-driven models such as PGPR [37] adapt these techniques to discover high-quality user-item paths that align with user preferences while offering interpretable rationales in the form of relation sequences.

The advantage of RL-based reasoning lies in its balance between exploration and exploitation: it learns to focus on semantically meaningful graph regions while maintaining interpretability via explicit reasoning paths. However, challenges remain in ensuring scalability to large-scale knowledge graphs and in avoiding spurious paths that achieve high rewards but offer weak semantic justification.

2.2.2. Rule learning & neural-symbolic methods

Rule learning provides a structured and symbolic way to capture regularities in knowledge graphs. Instead of relying purely on numerical embeddings, rule-based methods derive logical inference patterns. For example, rules of the form:

$$\textit{User} \xrightarrow{\textit{purchased}} \textit{ItemA}, \quad \textit{ItemA} \xrightarrow{\textit{similarTo}} \textit{ItemB} \Rightarrow \textit{User} \xrightarrow{\textit{interestedIn}} \textit{ItemB}.$$

These patterns explicitly encode semantic relationships that can directly support explainable recommendation.

Early work such as AMIE+ and AnyBURL [5, 15] focused on mining Horn rules from knowledge graphs, which have since been adapted to recommender contexts. Recent neural-symbolic approaches, such as NeuralLP [39] and DRUM [24], integrate differentiable rule learning with neural reasoning, enabling systems to jointly learn symbolic structures and embeddings. In KGRS, such methods allow combining the logical transparency of rules with the representational power of embeddings, bridging symbolic reasoning and sub-symbolic learning.

The main benefit of rule learning and neural-symbolic integration is faithful explainability: users can trace back recommendations to explicit, human-readable logical patterns. However, the interpretability advantage often comes at the cost of reduced scalability and generalization when compared to deep embedding-based architectures.

2.2.3. Counterfactual & Causal reasoning

Counterfactual and causal reasoning mechanisms have emerged as crucial tools for producing trustworthy and robust explainable recommendations. Instead of merely correlational signals, causal approaches seek to uncover why a recommendation occurs and how outcomes would change under different circumstances.

In the KGRS setting, causal inference frameworks such as structural causal models (SCMs) can be used to model user-item interactions as causal graphs, where interventions simulate alternative user behaviors or content exposures. For instance, counterfactual reasoning can answer questions like: *"If the user had not interacted with item A, would item B still have been recommended?"* This form of reasoning enhances both transparency and fairness by identifying spurious correlations and confounding effects.

Recent works such as CausalRec [20] integrate causal graph estimation with knowledge graph embeddings, disentangling direct and indirect influence pathways among entities. By modeling interventions explicitly, these systems support explanation generation through contrastive statements *(why A instead of B)* or hypothetical simulations *(what if X were changed)*. The integration of counterfactual reasoning into KGRS thus provides a foundation for faithful, actionable, and fairness-aware explanations.

2.2.4. Knowledge Graph Message Passing

Knowledge graph message passing refers to a family of graph-based learning mechanisms that propagate information between entities and relations through structured neighborhoods. This paradigm is primarily instantiated through Graph Neural Networks (GNNs) and their knowledge-graph-specific variants, which enable the representation of both node attributes and relational semantics in a unified learning process. In the context of KGRS, message passing facilitates the transfer of user preference signals and semantic dependencies across multi-hop connections, allowing the system to uncover indirect associations between users and items.

Early frameworks such as KGNN-LS [32] extended traditional GNNs by incorporating relation-specific transformations, enabling message propagation that respects heterogeneous relation types in knowledge graphs. Subsequent models, such as LightGCN [7], simplified the propagation process by removing non-linearities and transformation matrices, focusing purely on collaborative signal diffusion, which

significantly improved scalability for large-scale recommender systems. More advanced architectures, such as R-GCN [25], model relation-specific weight matrices to capture diverse semantic roles of edges, while CompGCN [30] introduces composition operators (e.g., circular-correlation, subtraction) to jointly encode entity and relation features.

From an explainability perspective, message passing provides a natural way to interpret how information flows through the knowledge graph. By tracing the aggregation paths or attention weights, one can visualize which neighboring entities and relations contributed most to a recommendation. Attention-based variants such as KGAT [31] explicitly learn importance weights over neighbors, thereby yielding interpretable rationales for why specific entities or relations were influential in the final recommendation decision.

Overall, message passing serves as a critical methodological foundation for KGRS, balancing representation learning and interpretability. It provides both high expressiveness for modeling graph-structured semantics and transparency through traceable signal propagation, forming a bridge between low-level embedding models and high-level reasoning frameworks.

2.3. Explainability mechanisms in KGRS

Explainability mechanisms in knowledge-graph-based recommender systems (KGRS) can be divided into model-specific and model-agnostic approaches. Model-specific methods exploit internal structures—such as reasoning paths, attention weights, or rules—to produce faithful explanations that reflect the model's actual decision process. Model-agnostic methods treat the system as a black box, generating explanations independently of its internal structure.

2.3.1. Model-specific explanations

Model-specific explanation methods leverage the internal structure or parameters of a particular KGRS model to generate explanations. In knowledge-graph-based recommenders, this often involves tracing reasoning paths, attention weights, or propagation signals that are inherent to the model's architecture.

For example, path-based models such as PGPR [37] or CAFE [36] produce explicit multi-hop paths connecting a user to recommended items. These paths serve as transparent explanations, as each entity and relation along the path can be interpreted as a semantic rationale for the recommendation. Similarly, embedding-based models with attention mechanisms, such as KGAT [31], can output attention scores over neighboring entities and relations, highlighting which connections contributed most to a recommendation. In neural-symbolic or rule-based methods, the mined rules or differentiable logical structures naturally provide human-readable reasoning chains, allowing the user to trace recommendations back to interpretable patterns [39, 24].

The advantage of model-specific explanations lies in their **faithfulness**: the explanation directly reflects the internal computation of the model, offering high alignment between the rationale and the actual decision process. However, these explanations are often tightly coupled with the model architecture and may not generalize across different types of KGRS.

2.3.2. Model-agnostic explanations

Model-agnostic explanation methods, in contrast, treat the KGRS as a black box and generate explanations independently of the internal structure. These approaches are useful when the model is too complex to interpret directly or when explanations are needed for multiple model types simultaneously.

Techniques such as LIME [23] or SHAP [12] can be adapted to KGRS by perturbing user or item inputs and observing changes in recommendation outputs, thereby estimating the contribution of different entities or features to the final decision. Another approach involves counterfactual explanations [20], where the system identifies minimal changes to user history or knowledge graph connections that would alter the recommendation outcome. This provides actionable insights, such as "Had the user not interacted with Item A, Item B would not have been recommended". Model-agnostic methods can also include surrogate models, such as decision trees or rule learners, trained to approximate the behavior of the original KGRS, enabling interpretable reasoning without exposing the original model internals.

The main benefit of model-agnostic explanations is **flexibility**: they can be applied to any KGRS regard-

less of architecture. However, there is often a trade-off in faithfulness, since the explanation may not perfectly reflect the exact reasoning process of the original model, and approximations or perturbations may introduce biases.

3

Methodology

This study aims to establish a systematic evaluation framework for path-level explainability in Knowledge Graph-based Recommender Systems (KGRS). Different representative models are compared between recommendation accuracy and different dimensions of explainability.

The proposed evaluation framework includes both standard recommendation metrics and explainability-specific metrics inspired by recent works. All experiments will be conducted on real-world datasets to ensure external validity.

3.1. Problem Definition

To formalize the task of time-aware knowledge graph reasoning for explainable recommendation, the notations used throughout this work are summarized in Table 3.1. These notations provide a consistent basis for defining the relevant entities, relations, and temporal features that are involved in reasoning over user–item interactions.

A knowledge graph (KG) is formally defined as a set of triples:

$$G_2 = \{(h, r, t) \mid h, t \in E, r \in R\},$$

where E denotes the set of entities, R denotes the set of relations, and each triple (h, r, t) represents a factual relation from head entity h to tail entity t.

The user-item interactions are represented as a bipartite graph:

$$G_1 = \{(u, y_{uv}, v) \mid u \in U, i \in V\},$$

where $y_{uv} = 1$ indicates an observed interaction between user u and item i, otherwise $y_{uv} = 0$.

Finally, the *Collaborative Knowledge Graph* G is constructed by integrating the user-item bipartite graph G_1 and the knowledge graph G_2:

$$G = \{(h, r, t) \mid h, t \in E', r \in R'\},$$

where $E' = E \cup U$ and $R' = R \cup \{\text{Interact}\}$, forming a unified graph for reasoning over both side information and historical interactions.

Figure 3.1 illustrates the overall structure of the Collaborative Knowledge Graph (CKG), which integrates the user-item interaction graph and the knowledge graph into a unified framework.

Although the formal definition of the Collaborative Knowledge Graph G is consistent across models, the actual construction varies slightly. Models such as PGPR, TPRec, and TMER only use the relations provided in the dataset (e.g., as shown in Table 3.2) to define user-item interactions, without explicitly adding additional *Interact* edges. In contrast, RippleNet and KGAT explicitly incorporate *Interact* relations by treating observed user-item purchases in the training set as additional edges in the graph.

Table 3.1: Notation and Annotation for Time-aware Knowledge Graph Reasoning [42].

Notations	Annotation
U	User set
V, I	Global item set
V_U	The observed interactions between users and items
$Y = \{y_{uv} \mid u \in U, v \in V\}$	User-item interaction matrix defined according to users' implicit feedback
$\hat{y}_{uv}$	The probability that user u will click item v
G_1	User-item bipartite graph
G_2	Knowledge graph containing side information
G	Collaborative Knowledge Graph, combining G_1 and G_2
N_h	The ego-network of entity h
V_U^T	The clustered time-aware interaction set, with L categories
G_T	Time-aware Collaborative Knowledge Graph
E'	The entities in Time-aware Collaborative Knowledge Graph
R_T	The relations in Time-aware Collaborative Knowledge Graph
T	The interaction timestamp set
$\hat{T}$	The recommend time
$\hat{V}$	The recommended item set
$\tau_{u,\hat{V}}$	The reasoning path set for user u to item set $\hat{V}$
$\mathcal{T}$	Temporal feature space, including statistical features and structural features
r_{kg}	The external knowledge graph relation set
$r_{interact}$	The time-aware interaction relation set
r	The embedding of relations
e_h, e_t	The embedding of entities
$W_t, W_{\hat{T}}$	The weight distribution of the Gaussian models at timestamp t, $\hat{T}$
W_u^h	The time cluster weight distribution of user u's interaction history
k, k', K	The k-th reasoning step, state history length k' and terminal step K
s_k	The state at step k
$a_k, \mathcal{A}_k$	The action and action space at step k
$g_R(\hat{v} \mid u)$	The time-aware based reward for recommended item $\hat{v}$ for user u
$\pi(\cdot \mid s_k, \mathcal{A}_k(u))$	The policy network at step k
$\hat{c}(s_k)$	The value network at step k

This allows these models to encode direct behavioral signals from the training data into the knowledge graph, enabling high-order propagation that captures both side information and user preferences.

Definition 1 (k-hop Path)**.** *A k-hop path from entity e_0 to entity e_k is defined as a sequence of $k+1$ entities connected by k relations in G_R:*

$$p_k(e_0, e_k) = \big(e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} \cdots \xrightarrow{r_k} e_k\big),$$

where each edge $e_{i-1} \xrightarrow{r_i} e_i$ corresponds to either $(e_{i-1}, r_i, e_i) \in G_R$ or $(e_i, r_i, e_{i-1}) \in G_R$, and $i \in [1, k]$.

A formal framework for knowledge graph reasoning in explainable recommendation can be established by defining the task of KG-based recommendation in terms of multi-hop paths.

Definition 2 (Knowledge Graph Reasoning for Explainable Recommendation (KGRE-Rec))**.** *Given a knowledge graph G_R, a user $u \in U$, and integers K and N, the objective is to identify a set of N recommended items*

$$\hat{V} = \{\hat{v}_n\}_{n=1}^{N} \subseteq V$$

such that for each pair $(u, \hat{v}_n)$, there exists a valid multi-hop reasoning path

$$\tau_{u,\hat{v}_n} = p_k(u, \hat{v}_n), \quad 2 \le k \le K,$$

which provides an interpretable explanation of why item $\hat{v}_n$ is recommended to user u. The set of all reasoning paths for user u is denoted as $\tau_{u,\hat{V}}$.

Time-awareness is introduced into the reasoning environment by incorporating interaction timestamps. Interaction events are grouped into L clusters through time-series clustering, and the interaction relation

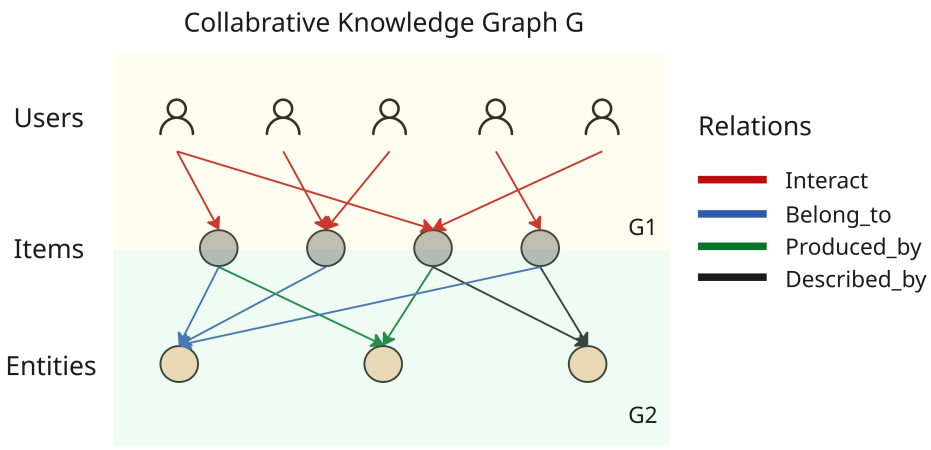


Figure 3.1: Illustration of the Collaborative Knowledge Graph (CKG)

V_U is correspondingly extended to

$$\{V_1^U, V_2^U, \ldots, V_L^U\}.$$

Definition 3 (Time-aware Collaborative Knowledge Graph (TCKG))**.** *A* Time-aware Collaborative Knowledge Graph (TCKG) *is obtained, which serves as the environment for time-aware path reasoning. The TCKG is formulated as*

$$G_T = \{(h, r, t) \mid h, t \in E',\ r \in R_T\},$$

where

$$R_T = R \cup \{V_1^U, V_2^U, \ldots, V_L^U\}.$$

By leveraging the TCKG, reasoning paths can be evaluated with temporal context, enabling recommendations that are both interpretable and aware of users' interaction dynamics over time.

Definition 4 (Meta-path)**.** *A meta-path[27] P in a network from entity v_0 to entity v_k is defined as a sequence of entities connected by relations:*

$$P : v_0 \xrightarrow{r_0} v_1 \xrightarrow{r_1} v_2 \xrightarrow{r_2} \cdots \xrightarrow{r_{k-1}} v_k,$$

where the composite relation from v_0 to v_k is given by

$$r = r_0 \circ r_1 \circ r_2 \circ \cdots \circ r_{k-1},$$

and $\circ$ denotes the composition operator on relations. A meta-path thus captures the semantic composite relation connecting v_0 and v_k along a predefined relational schema.

Definition 5 (User-Item Interaction Matrix)**.** *The user-item interaction matrix $Y = \{y_{uv} \mid u \in U, v \in V\}$ is defined according to users' implicit feedback:*

$$y_{uv} = \begin{cases} 1, & \text{if interaction } (u, v) \text{ is observed;} \\ 0, & \text{otherwise.} \end{cases}$$

A value of 1 for y_{uv} indicates there is an implicit interaction between user u and item v.

Definition 6 (Relevant Entity)**.** *Given the interaction matrix Y and knowledge graph G, the set of k-hop relevant entities for user u is defined as:*

$$\mathcal{E}_u^k = \{t \mid (h, r, t) \in G \text{ and } h \in \mathcal{E}_u^{k-1}\}, \quad k = 1, 2, \ldots, H,$$

where

$$\mathcal{E}_u^0 = V_u = \{v \mid y_{uv} = 1\}$$

is the set of user u's historically purchased items, which can be seen as the seed set of user u in the KG.

Definition 7 (Ripple Set)**.** *Given the relevant entities defined above, the k-hop ripple set of user u is defined as the set of knowledge triples starting from $\mathcal{E}_u^{k-1}$:*

$$S_u^k = \{(h, r, t) \mid (h, r, t) \in G \text{ and } h \in \mathcal{E}_u^{k-1}\}, \quad k = 1, 2, \ldots, H.$$

The ripple set represents the propagation of user u's interests through the knowledge graph along k hops.

3.2. Datasets

All experiments are performed on subdomains of the Amazon e-commerce datasets [14], which contain product reviews along with associated meta-information from Amazon.com. Three subdomains are considered in this study: Clothing, Cell Phones, and Beauty. Each subdomain is treated as an individual benchmark and represented as a knowledge graph comprising 5 types of entities and 7 types of relations. Table 3.2 summarizes the number of entities and relation statistics for each dataset.

Within these knowledge graphs, certain relations, particularly Mention and Described_by, are observed to dominate in frequency and are both linked to the Feature entity. Since these features often contain noisy or less informative words, a preprocessing step is applied to filter out less salient features. Specifically, feature words appearing more than 5,000 times or with a TF-IDF score below 0.1 are discarded to retain more meaningful attributes for reasoning.

For evaluation, 70% of each user's purchase interactions are used for training, while the remaining 30% form the test set. The objective of the knowledge graph reasoning and recommendation task is to recommend items purchased by users in the test set, accompanied by interpretable reasoning paths connecting the user and item within the knowledge graph.

Table 3.2: Descriptions and statistics of three Amazon e-commerce datasets: Clothing, Cell Phones, and Beauty [37].

Entities		**Number of Entities**		
Entity	**Description**	**Clothing**	**Cell Phones**	**Beauty**
User	User in recommender system	39,387	27,879	22,363
Item	Product to be recommended	23,033	10,429	12,101
Feature	Product feature word from reviews	21,366	22,493	22,564
Brand	Brand or manufacturer of the product	1,182	955	2,077
Category	Category of the product	1,193	206	248
Relations		**Number of Relations per Head Entity**		
Relation	**Description**	**Clothing**	**Cell Phones**	**Beauty**
Purchase	User $\xrightarrow{\text{purchase}}$ Item	7.08 ± 3.59	6.97 ± 4.55	8.88 ± 8.16
Mention	User $\xrightarrow{\text{mention}}$ Feature	440.20 ± 452.38	652.08 ± 1,335.76	806.89 ± 1,344.08
Described_by	Item $\xrightarrow{\text{described\_by}}$ Feature	752.75 ± 909.42	1,743.16 ± 3,482.76	1,491.16 ± 2,553.93
Belong_to	Item $\xrightarrow{\text{belong\_to}}$ Category	6.72 ± 2.15	3.49 ± 1.08	4.11 ± 0.70
Produced_by	Item $\xrightarrow{\text{produced\_by}}$ Brand	0.17 ± 0.38	0.52 ± 0.50	0.83 ± 0.38
Also_bought	Item $\xrightarrow{\text{also\_bought}}$ Item	61.35 ± 32.99	56.53 ± 35.82	73.65 ± 30.69
Also_viewed	Item $\xrightarrow{\text{also\_viewed}}$ another Item	6.29 ± 6.17	1.24 ± 4.29	12.84 ± 8.97
Bought_together	Item $\xrightarrow{\text{bought\_together}}$ another Item	0.69 ± 0.90	0.81 ± 0.77	0.75 ± 0.72

3.3. Evaluation Metrics

To evaluate both the predictive performance and explainability of the selected KGRS models, we employ two categories of metrics: traditional recommendation accuracy metrics and explainability-specific metrics.

3.3.1. Recommendation Metrics

To evaluate the basic recommendation accuracy of KGRS models, we adopt four standard top-K ranking metrics: Precision, Recall, Hit Rate (HR), and Normalized Discounted Cumulative Gain (NDCG). These metrics are defined as follows:

$$\text{Precision@K} = \frac{|TP|}{|TP| + |FP|}, \quad \text{Recall@K} = \frac{|TP|}{|TP| + |FN|},$$
$$\text{HR@K} = \frac{N_{\text{hits}}@K}{N_{\text{users}}}, \quad \text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}}.$$

Here, DCG@K represents the Discounted Cumulative Gain at position K, and IDCG@K denotes the Ideal Discounted Cumulative Gain through position K. $|TP|$ is the number of recommended items that are actually relevant at position K, $|TP| + |FN|$ is the total number of relevant items, and $|TP| + |FP|$ is the total number of recommended items within the top-K list.

As for HR@K, $N_{\text{hits}}@K$ denotes the number of users for whom at least one relevant item appears within the top-K recommendations, while N_{users} is the total number of users.

These metrics jointly measure the accuracy and ranking effectiveness of recommendation models, ensuring fair and consistent evaluation across different paradigms.

3.3.2. Explainability Metrics

To comprehensively evaluate both the *explainability quality* of models, we adopt metrics from related work. The evaluation is organized into three major categories as follows.

(1) Explanation-Ground Truth Consistency Metrics

Following [3], we evaluate the standard recommendation accuracy and explanation quality using **Recall**, **Precision**, and the **F1-score**. Their definitions originate from the work [3], which further refines the evaluation process by leveraging users' historical reviews as ground-truth rationales for item preferences.

Specifically, we adopt the same automatic evaluation strategy as TPRec [3], where each user's textual reviews are treated as the ground-truth reasons (G_u) explaining why the user interacted with or purchased specific items. For a given user u, and a recommended item $\hat{v} \in \hat{V}$ appearing within the top-K ranked list, the model generates reasoning paths $\tau_{u,\hat{V}}$ that connect the user and the item through entities in the knowledge graph. From these paths, all entity nodes that correspond to *word features* are extracted and organized as an explanation list:

$$S_u = [s_u^{(0)}, s_u^{(1)}, \ldots, s_u^{(r)}],$$

which represents the model's predicted explanations for user u.

Meanwhile, the user's review corpus is preprocessed to remove non-salient or overly common words. Specifically, words with frequency higher than 5000 and TF-IDF lower than 0.1 are filtered out to retain only meaningful review aspects. The resulting set of informative review terms forms the ground-truth explanation set:

$$G_u = [g_u^{(0)}, g_u^{(1)}, \ldots, g_u^{(n)}].$$

Thus, if a reasoning path contains more entities that overlap with the ground-truth words in G_u, the model achieves better explainability.

Finally, the metrics are computed as follows:

$$\text{Recall} = \frac{|S_u \cap G_u|}{|G_u| + 1}, \quad \text{Precision} = \frac{|S_u \cap G_u|}{|S_u| + 1}, \quad F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall} + 1}. \tag{3.1}$$

Here:

- S_u is the set of entities (words or attributes) used by the model to justify the recommendation;
- G_u is the set of ground-truth entities extracted from user reviews;
- The $+1$ term in the denominators prevents division by zero for users with no overlapping words or empty review sets.

Intuitively, **Recall** measures how many of the aspects that the user truly cared about (G_u) are captured by the model's generated explanations (S_u); **Precision** measures the proportion of the model's explanations that are actually relevant or liked by the user; **F1-score** represents the harmonic mean between Precision and Recall, providing a balanced overall evaluation of explanation accuracy.

Finally, to ensure a robust global assessment, we compute the average Recall, Precision, and F1-score across all users in the test set. These three metrics collectively quantify the *consistency between the model's generated explanations and the users' ground-truth reasoning*. Specifically, Recall reflects how well the model captures the aspects that users truly value in their reviews (coverage of genuine reasoning), while Precision indicates how accurately the generated explanations correspond to the user's real purchase motives (relevance of reasoning). The F1-score, as the harmonic mean of the two, represents a balanced evaluation of both coverage and relevance. Hence, higher scores across these metrics demonstrate that the explanations produced by the model are not only semantically aligned with users' authentic preferences but also faithfully reflect their actual decision-making rationales.

(2) Explanation Temporal, Popularity, and Diversity Metrics
To comprehensively evaluate the temporal relevance, personalization, and structural diversity of generated explanations, we adopt three path-level metrics proposed by [1]: **Linking Interaction Recency (LIR)**, **Shared Entity Popularity (SEP)**, and **Explanation Type Diversity (ETD)**. These metrics jointly measure how well a model balances recency sensitivity, popularity bias reduction, and reasoning diversity within explanation paths derived from the knowledge graph.

(a) Linking Interaction Recency (LIR). The *Linking Interaction Recency* (LIR) metric quantifies the temporal freshness of user interactions that appear in the explanation paths, thereby capturing the extent to which recent user behaviors contribute to the recommendation rationale. This metric reflects the assumption that recent interactions are more indicative of a user's current preferences than older ones, and explanations grounded in temporally close interactions are perceived as more relevant and trustworthy.

Formally, given a user $u \in U$ and the set of products P_u that the user has interacted with, we denote the chronologically ordered list of interactions as:

$$P_u = [(v_1, t_1), (v_2, t_2), \ldots, (v_{|P_u|}, t_{|P_u|})],$$

where $v_i \in P_u$ represents an interacted product, and $t_i \in \mathbb{N}$ is the timestamp of the interaction, with $t_i \leq t_{i+1}$ for all $i = 1, \ldots, |P_u|$.

To compute how recently an interaction in the explanation path occurred, we apply an *exponentially weighted moving average (EWMA)* over the interaction timestamps in P_u. For each interaction (v_i, t_i), the LIR score is defined recursively as:

$$LIR(p^i, t^i) = (1 - \beta_{LIR}) \cdot LIR(p^{i-1}, t^{i-1}) + \beta_{LIR} \cdot t^i, \tag{3.2}$$

where $\beta_{LIR} \in [0, 1]$ is a temporal decay factor that controls how much weight is assigned to the most recent timestamp. This recursive formulation ensures that newer interactions have exponentially greater influence on the final LIR value compared to older ones.

(b) Shared Entity Popularity (SEP). The *Shared Entity Popularity* (SEP) metric quantifies the extent to which entities connecting users' past interactions and recommended items are globally popular or relatively niche within the knowledge graph. This property serves to evaluate the presence of *popularity bias* in explanation paths, since over-reliance on popular entities—such as well-known brands, mainstream actors, or high-frequency categories—can lead to generic and less personalized recommendations.

Formally, we assume that the number of relationships in which a shared entity participates within the knowledge graph (*i.e.*, its in-degree) serves as a proxy for its popularity. For example, the popularity of an actor entity can be measured by counting how many movies the actor has appeared in, while the popularity of a brand can be estimated by how many products are associated with it.

Let E_λ denote the list of entities of a given type λ (e.g., actor, brand, category) in the knowledge graph, sorted by their popularity scores:

$$E_\lambda = [(e_1, v_1), (e_2, v_2), \ldots, (e_{|E_\lambda|}, v_{|E_\lambda|})],$$

where $e_i \in E_\lambda$ is an entity of type λ, and $v_i \in \mathbb{N}$ represents the number of relationships (edges) that entity e_i participates in, satisfying $v_i \leq v_{i+1}$ for all i.

To compute the popularity of shared entities that appear in an explanation path, we apply an *exponential decay* over their global popularity scores to emphasize less frequent entities. The SEP value for a shared entity e^i involved in a reasoning path is defined recursively as:

$$SEP(e^i, v^i) = (1 - \beta_{SEP}) \cdot SEP(e^{i-1}, v^{i-1}) + \beta_{SEP} \cdot v^i, \tag{3.3}$$

where $\beta_{SEP} \in [0, 1]$ is a smoothing factor that controls how much the popularity of the current entity influences the overall score.

(c) Explanation Type Diversity (ETD). The *Explanation Type Diversity* (ETD) metric measures the semantic richness and structural diversity of the reasoning paths used to generate explanations for each user. A high ETD value indicates that the recommender provides varied, non-repetitive justification patterns (e.g., mixing "same actor" and "directed by" reasoning), which enhances interpretability and avoids monotonous or redundant explanations.

Formally, a reasoning path is defined as an alternating sequence of entities and relationships:

$$l = \{e_1, r_1, e_2, r_2, \ldots, e_{|n|-1}, r_{|n|}, e_{|n|}\},$$

where each e_i is an entity (e.g., a user, clothing item, or brand) and each r_i represents a relationship between entities.

For example, in the fashion domain, a possible explanation path could be:

$$\text{User}_{u_1} \xrightarrow{purchase} \text{ClothingItem}_{c_1} \xrightarrow{produced\_by} \text{Brand}_{b_1} \xleftarrow{produced\_by} \text{ClothingItem}_{c_2}.$$

The **path pattern** of l, denoted as ω_l, is defined by the ordered sequence of relationships appearing in the path:

$$\omega_l = \{r_1 \circ r_2 \circ \cdots \circ r_{|\omega_l|} \mid r_i \in l,\ i \leq |l|\}.$$

In the above example, the path pattern is *purchase* $\circ$ *produced_by* $\circ$ *also_bought*. The **path type**, denoted as τ_l, is determined by the last relationship $r_{|\omega_l|}$ in the pattern (e.g., $\tau_l =$ *also_bought* in the example).

Given a user u, let $\tilde{L}_u$ denote the set of top-k explanation paths generated for that user, and let $\omega_{\tilde{L}_u}$ represent the set of distinct path patterns found among them. The ETD score is defined as:

$$ETD(\tilde{L}_u) = \frac{|\omega_{\tilde{L}_u}|}{\min(k, |\omega_L|)}, \tag{3.4}$$

where ω_L is the global set of all possible path patterns in the knowledge graph.

This ratio measures the degree of semantic diversity in the explanation set.

Interpretation. Together, these three metrics provide a holistic view of explanation quality:

- **LIR** assesses the *temporal validity* of explanations, ensuring recency-aligned reasoning.
- **SEP** measures the *personalization level*, rewarding explanations that avoid over-reliance on popular entities.
- **ETD** captures the *diversity of reasoning*, encouraging multiple distinct semantic paths for interpretability.

Higher scores across these metrics indicate explanations that are more timely, personalized, and varied, thus improving both the informativeness and trustworthiness of the recommendation rationale.

(3) Explanation Faithfulness

Following [43], **faithfulness** is measured using the Jensen–Shannon (JS) divergence between the model's *training rule distribution* $F(u)$ (the distribution over logic rules learned during training) and its *generated rule distribution* (the distribution over rules inferred in the explanations). This measurement evaluates the consistency between the generated explanations and the model's actual reasoning process.

Training Rule Distribution. For each user u, approximately 1,000 paths between the user node and its connected item nodes are sampled from the training graph. Based on these sampled paths, the corresponding logic rules are extracted and the empirical rule distribution is computed, denoted as $F(u)$. Specifically, each path is mapped to a rule according to its relation sequence, which corresponds to the path pattern defined in Section 3.3 (i.e., the ordered sequence of relationships in the path). The frequency of each rule is normalized to form a probability distribution. This distribution reflects the reasoning patterns that the model is exposed to during training and thus approximates its intrinsic reasoning preference.

Generated Rule Distributions. During the test phase, for each selected user, the model generates explainable paths for recommended items. Two types of generated rule distributions are constructed:

- **Frequency-based distribution** $Q_f(u)$: All generated explanation paths for user u are collected, and the corresponding logic rules are extracted. The empirical frequency of each rule is then normalized to form a probability distribution. In this setting, rules that appear more frequently across recommended items receive higher probability mass. This version emphasizes the global structural patterns exhibited in the generated explanations.
- **Weight-based distribution** $Q_w(u)$: Instead of counting rule frequency uniformly, each explanation path is weighted according to the model's internal attention score or confidence score assigned during aggregation. The contribution of a rule is proportional to the cumulative weight of the paths associated with that rule. The weighted counts are then normalized to obtain a probability distribution. This distribution captures not only how often a rule appears, but also how important the model considers that rule during decision making.

Faithfulness Metrics. The faithfulness scores are computed as:

$$JS_f = \mathbb{E}_{u\sim\mathcal{U}}\left[D_{JS}\big(Q_f(u) \parallel F(u)\big)\right], \quad JS_w = \mathbb{E}_{u\sim\mathcal{U}}\left[D_{JS}\big(Q_w(u) \parallel F(u)\big)\right], \tag{3.5}$$

where $D_{JS}(\cdot \parallel \cdot)$ denotes the Jensen–Shannon divergence and $\mathcal{U}$ is the set of evaluated users.

Difference Between JS_f and JS_w. The key distinction lies in how rule importance is modeled:

- JS_f evaluates structural faithfulness at the *occurrence level*. It measures whether the types of rules appearing in the generated explanations match those observed during training, regardless of their relative importance in prediction.
- JS_w evaluates faithfulness at the *importance-weighted level*. By incorporating internal attention or confidence weights, it assesses whether the model assigns importance to explanation rules in a manner consistent with its learned reasoning distribution.

Therefore, JS_f reflects distributional similarity in terms of rule coverage, while JS_w provides a stricter evaluation that accounts for the model's internal prioritization mechanism.

A smaller value of JS_f indicates that the generated explanations cover similar reasoning patterns to those observed during training. A smaller value of JS_w further indicates that the relative importance assigned to those reasoning patterns is also consistent with the model's intrinsic decision process.

Table 3.3: Weight types used in computing the weighted generated rule distribution $Q_w(u)$ for different models.

Model	Weight Type for $Q_w(u)$
PGPR	Path selection probability (from policy)
TPRec	Path selection probability (from policy)
TMER	Attention score over reasoning paths
RippleNet	Relevance probability of item nodes
KGAT	Attention score over reasoning paths

Weight Types for Generated Rule Distributions. To clarify how the weighted rule distribution $Q_w(u)$ is computed for different models, Table 3.3 summarizes the type of weight used for each model. This helps interpret JS_w in the context of each model's internal scoring mechanism.

This table clarifies that while PGPR and TPRec rely on the policy-derived probability of path selection, TMER and KGAT use attention scores to weight paths, and RippleNet weights paths according to item relevance probabilities. These differences explain why JS_w may vary even for models with similar JS_f, as JS_w additionally reflects how the model prioritizes reasoning rules internally.

3.4. Models

To comprehensively evaluate path-level explainability, five representative KGRS models are chosen, each corresponding to a distinct methodological paradigm. Collectively, these models cover embedding-based representation learning, reinforcement-guided reasoning, temporal path modeling, and hybrid reasoning that integrates knowledge graph embeddings with multi-hop preference propagation.

Table 3.4: Representative models used for comparative evaluation

Paradigm	Model	Key Feature
Embedding-based	**TPRec** [42]	Time-aware embeddings integrating interaction recency into representation learning.
Embedding-based	**PGPR** [37]	Policy-guided path reasoning through reinforcement learning.
Embedding-based	**KGAT** [31]	Knowledge graph attention network leveraging graph structure for recommendation with relational reasoning.
Path-based	**TMER** [3]	Temporal meta-path reasoning with explainable sequential patterns.
Hybrid	**RippleNet** [33]	Iterative propagation of user preferences along the knowledge graph to capture multi-hop semantic relationships.

These five models represent the major methodological categories within KGRS: embedding-based propagation (TPRec, PGPR, KGAT), explicit path-based reasoning (TMER), and hybrid multi-hop reasoning combining embeddings and path information (RippleNet). By covering these paradigms, the influence of structural design choices on both recommendation accuracy and path-level explainability can be systematically evaluated.

3.4.1. Reinforcement Learning Methods

In the context of knowledge graph based explainable recommendation, reinforcement learning (RL) provides a principled framework to model multi-hop reasoning as a sequential decision process. Two representative RL-based methods are Policy-Guided Path Reasoning (PGPR) and Time-aware Path Reasoning for Recommendation (TPRec). PGPR conducts explicit path search in a static knowledge graph, while TPRec further incorporates temporal interaction patterns by augmenting the graph with time-aware relations.

Policy Guided Path Reasoning (PGPR)

Policy-Guided Path Reasoning (PGPR) casts the path reasoning problem as a Markov decision process (MDP) in which an agent learns to traverse a knowledge graph from a user node to reachable item

nodes. A policy is learned such that the induced paths not only connect a user to recommended items but also serve as explanations for those recommendations.

Formally, PGPR defines the following MDP components:

State At timestep t, the state is defined as

$$s_t = (u, e_t, h_t), \tag{3.6}$$

where u is the fixed user entity, e_t is the current entity reached by the agent in the KG, and h_t encodes the history of the reasoning path up to t. This history typically includes the sequence of visited entities and relations, enabling the policy to be conditioned on the past trajectory.

Action The action at state s_t is selected from a user-pruned action space $\tilde{\mathcal{A}}_t(u)$, defined as the set of outgoing edges from e_t that exclude entities and relations already in the history. An action is represented as a relation–entity pair

$$a_t = (r_{t+1}, e_{t+1}), \tag{3.7}$$

meaning the agent will traverse relation r_{t+1} to reach entity e_{t+1}. To control the combinatorial explosion of edges, a user-conditional pruning strategy is used to restrict the action space.

Transition Due to the structural properties of the knowledge graph, a state is fully determined by the current entity position and the path history. Given the current state $s_t = (u, e_t, h_t)$ and the selected action $a_t = (r_{t+1}, e_{t+1})$, the transition to the next state is deterministic and defined as:

$$\mathbb{P}\left(s_{t+1} = (u, e_{t+1}, h_{t+1}) \mid s_t = (u, e_t, h_t),\ a_t = (r_{t+1}, e_{t+1})\right) = 1 \tag{3.8}$$

where the updated history is $h_{t+1} = h_t \cup \{(r_{t+1}, e_{t+1})\}$.

This deterministic transition reflects the fact that once an outgoing edge (r_{t+1}, e_{t+1}) is selected from entity e_t, the next entity position is uniquely determined by the graph structure.

Reward PGPR uses a soft terminal reward defined only at the end of a path of fixed length T. Let $f(u, i)$ be a scoring function that estimates the relevance of item i to user u. The terminal reward is defined as

$$R_T = \begin{cases} \frac{\max(0,\, f(u, e_T))}{\max_{i \in V} f(u,i)}, & \text{if } e_T \in V, \\ 0, & \text{otherwise}, \end{cases} \tag{3.9}$$

where V is the item set. This reward lies in $[0, 1]$ and encourages the agent to end at high-relevance items, rather than simply reaching arbitrary entities. Intermediate rewards are zero, making the reward sparse and focused on final outcomes.

Scoring Function To guide action pruning and reward computation, PGPR introduces a multi-hop scoring function based on relation patterns. A key property of the knowledge graph G_R is that given the type of a head entity and a valid relation, the type of the tail entity is uniquely determined. This property can be extended to a chain rule over entity-relation types, forming a sequence $\{e_0, r_1, e_1, r_2, \ldots, r_k, e_k\}$. If the type of e_0 and relations $\{r_1, \ldots, r_k\}$ are specified, then the types of intermediate entities $\{e_1, \ldots, e_k\}$ are uniquely determined.

Based on this property, a k-hop *pattern* is defined as a sequence of relations $\tilde{r}_k = \{r_1, \ldots, r_k\}$ that forms a valid path between two entities. Considering that reverse edges are allowed in reasoning, each relation in a pattern can be either forward or backward. A special case, termed the *1-reverse k-hop pattern*, denoted as $\tilde{r}_{k,j}$ ($j \in [0, k]$), consists of j forward relations followed by $k - j$ backward relations.

Given a 1-reverse k-hop pattern $\tilde{r}_{k,j}$, the similarity between two entities e_0 and e_k is quantified by the following scoring function:

$$f(e_0, e_k \mid \tilde{r}_{k,j}) = \left\langle e_0 + \sum_{s=1}^{j} r_s,\ e_k + \sum_{s=j+1}^{k} r_s \right\rangle + b_{e_k} \tag{3.10}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product, e and r are d-dimensional embeddings of entities and relations, and b_{e_k} is a bias term associated with entity e_k.

This formulation generalizes several special cases. When $k = 0$, the scoring function reduces to the similarity between two entity embeddings:

$$f(e_0, e_k \mid \tilde{r}_{0,0}) = \langle e_0, e_k \rangle + b_{e_k}.$$

When $k = 1$ and $j = 1$, the scoring function degenerates to the translational embedding form (e.g., TransE):

$$f(e_0, e_k \mid \tilde{r}_{1,1}) = \langle e_0 + r_1, e_k \rangle + b_{e_k}.$$

For $k \geq 1$ and $1 \leq j \leq k$, the general form measures entity similarity under a 1-reverse multi-hop relational pattern. This scoring function is used to evaluate candidate actions during pruning and to define the reward signal in reinforcement learning.

Time aware Path Reasoning for Recommendation (TPRec)
Time-aware Path Reasoning for Recommendation (TPRec) extends PGPR to incorporate temporal interaction information, explicitly modeling the fact that user-item interactions occur at specific timestamps. These temporal patterns are crucial because user preferences evolve over time, and the reasoning paths used for explanation should reflect both semantic and temporal relevance.

Time-aware Collaborative Knowledge Graph (TCKG). TPRec constructs a Time-aware Collaborative Knowledge Graph (TCKG) by clustering interaction timestamps into L discrete time categories. Each user-item interaction is then mapped to a time-aware relation according to its corresponding time cluster, resulting in the extended relation set $\{V_1^U, V_2^U, \ldots, V_L^U\}$. This approach ensures that the temporal component is explicitly represented in the graph, allowing the reinforcement learning agent to consider not only structural connectivity but also time-sensitive context when exploring reasoning paths.

Temporal Feature Extraction. To better capture temporal dynamics, TPRec derives both statistical and structural time features from user interactions. Statistical features include explicit attributes such as year, month, season, or event-based categories (e.g., holidays or sales periods). Structural features capture patterns in user activity over time, such as recency trends or periodicity in interactions. Incorporating these features into the TCKG ensures that reasoning paths reflect temporally coherent sequences, enhancing the relevance and interpretability of explanations.

Time-aware Path Exploration and Reward Mechanism. During path reasoning, TPRec employs a reinforcement learning agent to explore candidate paths between users and items. Unlike traditional PGPR, TPRec integrates temporal consistency into the reward function. Specifically, paths that traverse interactions temporally close to the user's current interests receive higher rewards, encouraging the model to prioritize recent or seasonally relevant behaviors. This mechanism ensures that generated reasoning paths are not only semantically meaningful but also temporally aligned, improving both recommendation accuracy and explanation fidelity.

Temporal-aware Embedding Learning. In addition to constructing a time-sensitive graph, TPRec learns embeddings for entities and relations that encode temporal information. Because each relation in the TCKG reflects a specific time category, the learned embeddings inherently capture both structural and temporal properties. These embeddings are then used in scoring functions and path selection, allowing the model to balance structural connectivity with temporal relevance when recommending items.

State, Action, and Transition Importantly, TPRec preserves the same MDP formulation as PGPR. That is, the definitions of state, action, and transition follow Equations 3.6-3.8 in the PGPR formulation.

Reward Unlike traditional reinforcement learning settings, there is no pre-defined target item for each user in recommendation, especially in the time-aware explainable scenario. Therefore, TPRec does not employ a binary reward indicating whether a specific item has been reached. Instead, it adopts a soft reward defined only at the terminal state $s_T = (u, e_T, h_T)$.

The terminal reward is defined as

$$R_T = \frac{g_R(e_T \mid u)}{\max_{v \in V} g_R(v \mid u)}, \tag{3.11}$$

where $e_T \in V$ is the predicted item reached at the end of the reasoning path, and V denotes the item set. The reward value is normalized into the range $[0, 1]$. Intermediate rewards are zero. This design encourages the agent to end at items with high time-aware relevance to the user.

Time-aware Scoring Function The terminal reward is determined by a time-aware scoring function $g_R(v \mid u)$. A commonly used multi-hop scoring function in static graphs is

$$g(v \mid u) = \langle \mathbf{e}_u + \mathbf{r}_{interact}, \mathbf{e}_v \rangle + b_v, \tag{3.12}$$

where $\mathbf{e}_u$ and $\mathbf{e}_v$ are user and item embeddings, $\mathbf{r}_{interact}$ is the interaction relation embedding, and b_v is the bias term of item v. This formulation assumes that if a multi-hop path connects u and v, it is reasonable to infer a potential interaction relation between them.

However, in the Time-aware Collaborative Knowledge Graph (TCKG), there exist multiple time-aware interaction relations $V_U^T = \{V_U^1, V_U^2, \ldots, V_U^L\}$, and it is unclear which one should be used for a given user. To address this issue, TPRec constructs a personalized time-aware interaction relation for each user based on her historical interactions:

$$\mathbf{r}_u^T = \mathbf{W}_{h_u} V_U^T, \tag{3.13}$$

where $\mathbf{W}_{h_u} = [w_{h_u}^1, w_{h_u}^2, \ldots, w_{h_u}^L]^\top$ is the time-cluster weight vector derived from user history $h_u = \{v_u^1, v_u^2, \ldots, v_u^q\}$. The weight for the l-th time cluster is computed as

$$w_{h_u}^l = \frac{\sum_{i=1}^{q} \mathbb{I}(v_u^i = V_U^l)}{q}, \tag{3.14}$$

where $\mathbb{I}(\cdot)$ is the indicator function. A larger weight indicates that the corresponding time cluster appears more frequently in the user's history.

Finally, the personalized time-aware scoring function is defined as

$$g_R(v \mid u) = \langle \mathbf{e}_u + \mathbf{r}_u^T, \mathbf{e}_v \rangle + b_v. \tag{3.15}$$

Compared with PGPR, whose reward is based solely on structural useritem relevance, TPRec incorporates temporal preference modeling through a personalized time-aware interaction relation, thereby encouraging paths that are both structurally meaningful and temporally plausible.

Training and Recommendation Both PGPR and TPRec are trained to maximize the expected cumulative reward along reasoning paths starting from a user node. Let the policy network be denoted as $\pi_\theta(a_t \mid s_t)$, mapping a state s_t to a probability distribution over valid actions $a_t \in \tilde{\mathcal{A}}_t(u)$. The value network $\hat{v}_\phi(s_t)$ estimates the expected return from state s_t and is used as a baseline to reduce variance in policy gradient updates. The objective is to maximize the expected discounted reward:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\Big[G_t \,\Big|\, s_0 = (u, u, \emptyset)\Big], \quad G_t = \sum_{k=0}^{T-1} \gamma^k R_{t+k+1}, \tag{3.16}$$

where $\gamma \in [0, 1]$ is the discount factor and R_{t+k+1} is the reward at timestep $t + k + 1$.

Policy parameters are updated via the *REINFORCE with baseline*[28]. During inference, candidate reasoning paths are generated by sampling actions from $\pi_\theta(a_t \mid s_t)$ or by using beam search to select top-K sequences that maximize cumulative path probabilities and rewards. For each candidate user-item pair (u, i), the path with the highest generative probability is selected as the interpretable reasoning path, and items are ranked according to $R(p_T(u, i))$ to produce the final recommendation list.

Comparsion of PGPR and TPRec

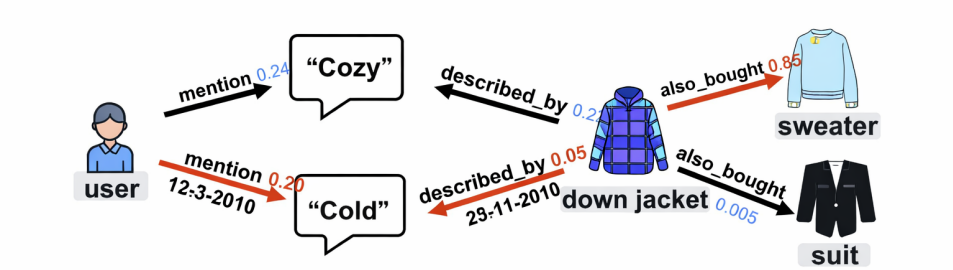


Figure 3.2: Case study comparison between the PGPR and TPRec models[42]

Figure 3.2 shows a recommendation scenario. By leveraging temporal information, TPRec can infer that it is winter and identify the most relevant contextual word as ”cold”. This allows TPRec to recommend more appropriate items, such as a ”sweater”. In contrast, PGPR, which does not utilize temporal information, cannot capture this seasonal context and may fail to suggest the most relevant item. This case study highlights how incorporating temporal knowledge enables TPRec to produce more context-aware and accurate recommendations compared to PGPR.

3.4.2. Temporal Meta-path Guided Methods

Temporal Meta-path Guided Explainable Recommendation (TMER) is a sequential explainable recommendation framework designed to leverage meta-path instances to model both structural context and temporal dynamics in user behavior on a dynamic knowledge graph. Relying solely on user-item meta-paths is restrictive for explainable recommendation, as such paths primarily reflect a user's general shopping interests without capturing the nuanced dependencies between items. In contrast, item-item meta-paths provide richer expressiveness by exploring higher-order relations among items. These relations can encode complementary products (e.g., a shirt and matching pants), substitutable items for previously purchased or viewed products (e.g., a red sweater → a blue sweater → a cardigan), and co-purchased products observed from social or historical interactions. Furthermore, item-item meta-paths naturally capture sequential dependencies between consecutively purchased items, thereby enhancing the modeling of both short-term and long-term sequential behaviors. By explicitly incorporating item-item meta-paths, TMER allows multiple underlying motivations to be reflected in item representations, improving both recommendation accuracy and explainability while effectively modeling temporal patterns in user interactions.

Meta-path Predefinition TMER differentiates meta-path instances into two semantic categories and uses them to extract contextual features:

- **User-Item Meta-paths:** These paths model direct and multi-hop relations between a user U and an item I through attribute and bridging entities:

$$\begin{aligned}
&\text{UIBI: } U \to I \to B \to I,\\
&\text{UICI: } U \to I \to C \to I,\\
&\text{UIBICI: } U \to I \to B \to I \to C \to I,\\
&\text{UICIBI: } U \to I \to C \to I \to B \to I,
\end{aligned}$$

  where B and C denote item brand and category entities respectively.

- **Item-Item Meta-paths:** These paths capture higher-order relations between consecutive items in user sequences, representing semantic and temporal dependencies:

$$\begin{aligned}
&\text{ICIBI: } I \to C \to I \to B \to I,\\
&\text{IBICI: } I \to B \to I \to C \to I,\\
&\text{ICICI: } I \to C \to I \to C \to I,\\
&\text{IBIBI: } I \to B \to I \to B \to I,\\
&\text{IUIUI: } I \to U \to I \to U \to I,\\
&\text{ICIUI: } I \to C \to I \to U \to I,\\
&\text{IBIUI: } I \to B \to I \to U \to I,
\end{aligned}$$

Overall Architecture The TMER architecture consists of four key stages:

1. **User and Item Embedding Initialization.** Users and items are first embedded via random walk based methods such as DeepWalk[17] to capture global structural information in the bipartite user-item graph. DeepWalk effectively learns co-occurrence relations among entities in paths, which benefits downstream meta-path context modelling.

2. **Meta-path Instance Mining and Sampling.** Instead of using all possible meta-path instances (which can be combinatorially large), TMER performs similarity-guided sampling: for a given meta-path schema, candidate next nodes are ranked by similarity to the current node, enabling the extraction of high-quality user-item and item-item path instances. This ensures representative reasoning paths are selected for each pair without overwhelming computation.

3. **Meta-path Context Encoding.** Sampled path instances are regarded as "sentences" with nodes as "tokens". Path embeddings are learned via a Word2Vec[16] style embedding followed by a multi-head sel-attention mechanism to fuse multiple instances into concise context vectors:

$$\text{Attention}(Q_\phi, K_\phi, V_\phi) = \text{softmax}\left(\frac{Q_\phi K_\phi^\top}{\sqrt{d_k}}\right) V_\phi, \tag{3.17}$$

$$\text{MultiHead}(Q_\phi, K_\phi, V_\phi) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W_O, \tag{3.18}$$

where Q_ϕ, K_ϕ, V_ϕ are the query, key and value matrices associated with path ϕ, d_k is the dimension of key vectors and W_O collects output weights. This self-attention captures intra-path dependencies among multiple meta-path instances.

4. **Temporal Sequence Modelling and Item Representation Update.** Instead of using recurrent networks to model temporal effects, TMER employs an item attention module using both previous item embedding and the corresponding item-item path context. The item-level update is conducted with multi-layer interactions:

$$h_i^{(1)} = \text{ReLU}(W_i^{-1} h_{i-1} + W_{\phi_{i-1\to i}}^{(1)} h_{\phi_{i-1\to i}} + b_i^{(1)}) \odot h_{i-1}, \tag{3.19}$$

$$h_i^{(2)} = \text{ReLU}(W_i h_i + W_{\phi_{i-1\to i}}^{(2)} h_{\phi_{i-1\to i}} + b_i^{(2)}) \odot h_i, \tag{3.20}$$

where h_{i-1} and h_i are consecutive item embeddings, $h_{\phi_{i-1\to i}}$ is the embedding of the item-item meta-path instance, and W, b are learned parameters of attention layers. For the first item in a user sequence,

its update also incorporates the user–item path instance:

$$h_{i=1} = \mathrm{ReLU}(W_i h_i + W_{\phi_{u\to i}} h_{\phi_{u\to i}} + b_i) \odot h_i. \tag{3.21}$$

This formulation allows the model to capture both short-term (last bought item) and long-term sequential dependencies in a unified attention style, avoiding heavier RNN[11]/LSTM[35, 18] architectures.

Training and Recommendation After the item representations $h_i^{(1)}$ and $h_i^{(2)}$ have been updated by the item attention module incorporating item-item meta-path context, the final user-item prediction scores are generated using a Multilayer Perceptron (MLP). The MLP receives concatenated embeddings of the user, the item, and relevant path instances.

User-Item Feature Concatenation: For each item i interacted by user u, the following vectors are concatenated into a single representation:

$$h_{u,i} = [h_u; h_i^{(1)}; h_i^{(2)}], \tag{3.22}$$

where $[\,;\,]$ denotes vector concatenation, h_u is the user embedding, and $h_i^{(1)}$, $h_i^{(2)}$ are the item embeddings updated by the attention mechanism capturing item-item meta-path context.

For the first item in a user sequence, which lacks a previous item, the user-item meta-path instance embedding is used:

$$h_{u,i=1} = [h_u; h_{\phi_{u\to 1}}; h_{i=1}], \tag{3.23}$$

where $h_{\phi_{u\to 1}}$ represents the embedding of the meta-path instance connecting the user u to the first item.

Rating Prediction: The concatenated vector is fed into a two-hidden-layer MLP to produce the predicted score:

$$r_{u,i} = \mathrm{MLP}(h_{u,i}), \tag{3.24}$$

with ReLU activation in the hidden layers and a sigmoid function in the output layer to produce scores in $[0, 1]$. A tower structure is applied in the MLP, reducing the layer size by half for each successive higher layer to capture higher-order abstractions efficiently.

Loss Function with Implicit Feedback: Negative sampling is adopted to handle implicit feedback. For each observed interaction (u, i) and a negative sample j drawn from a uniform noise distribution P_{neg}, the loss is defined as:

$$\mathrm{loss}_{u,i} = -\mathbb{E}_{j\sim P_{\text{neg}}} \log(1 - r_{u,j}), \tag{3.25}$$

encouraging higher scores for observed interactions and lower scores for unobserved ones. The overall training objective minimizes the sum of this loss across all user-item interactions in the dataset.

This formulation allows joint modeling of user preferences, item-item meta-path dependencies, and user-item meta-path instances, enabling accurate and explainable sequential recommendations without recurrent networks.

3.4.3. Preference Propagation Methods

RippleNet is an end-to-end knowledge-graph-aware recommender system that propagates user preferences over a knowledge graph to predict click probabilities. Unlike traditional collaborative filtering (CF) methods, which learn latent representations of users and items and predict interactions by directly applying similarity functions such as inner product, RippleNet leverages the structure of the knowledge graph to model fine-grained user-item interactions.

RippleNet is considered a hybrid approach. Traditional **embedding-based methods** represent entities and relations as low-dimensional vectors to capture semantic similarity, but they often ignore the explicit relational paths in the knowledge graph. **Path-based methods**, on the other hand, explicitly explore meta-paths or relation sequences between users and items to model connectivity, but they typically

rely on hand-crafted paths and cannot fully capture latent semantic information. RippleNet unifies the advantages of both approaches: it uses embeddings for entities and relations to encode latent features (like embedding-based methods), while recursively propagating user preferences along multiple hops in the knowledge graph (analogous to path-based reasoning), effectively discovering hierarchical and multi-hop relationships without manually designing paths. This combination of embedding representation and multi-hop preference propagation is why RippleNet is categorized as a **hybrid method**.

Each item v is represented by an embedding vector $\mathbf{v} \in \mathbb{R}^d$, which may incorporate one-hot IDs, item attributes, bag-of-words features, or contextual information depending on the application scenario. A user's historically interacted items serve as seeds in the knowledge graph. Denote S_u^1 as the 1-hop ripple set of user u, containing all triples (h_i, r_i, t_i) connected to the user's history. Each triple is assigned a relevance probability with respect to the target item v, calculated as:

$$p_i = \text{softmax}(\mathbf{v}^\top R_i \mathbf{h}_i) = \frac{\exp(\mathbf{v}^\top R_i \mathbf{h}_i)}{\sum_{(h,r,t)\in S_u^1} \exp(\mathbf{v}^\top R \mathbf{h})}, \tag{3.26}$$

where $\mathbf{h}_i$ and R_i are the embeddings of the head entity and the relation, respectively. This probability measures the similarity between item v and entity h_i in the relation space.

The first-order response vector of user u is computed as the weighted sum of tail embeddings:

$$\mathbf{o}_u^1 = \sum_{(h_i,r_i,t_i)\in S_u^1} p_i \, \mathbf{t}_i, \tag{3.27}$$

where $\mathbf{t}_i$ is the embedding of the tail entity. Conceptually, this is analogous to item-based CF, in which a user's representation is derived from related items rather than an independent feature vector.

Preference propagation is performed iteratively over higher-order ripple sets S_u^i ($i = 2, \ldots, H$) by replacing the input embedding with the previous response vector. The final user embedding with respect to item v is obtained by aggregating all responses:

$$\mathbf{u} = \mathbf{o}_u^1 + \mathbf{o}_u^2 + \cdots + \mathbf{o}_u^H. \tag{3.28}$$

Finally, the predicted click probability of user u on item v is calculated by a sigmoid over the inner product of the user and item embeddings:

$$\hat{y}_{uv} = \sigma(\mathbf{u}^\top \mathbf{v}), \quad \sigma(x) = \frac{1}{1+e^{-x}}. \tag{3.29}$$

This recursive propagation allows RippleNet to capture hierarchical and latent interests of users in the knowledge graph, automatically discovering multi-hop relations without any hand-crafted paths, thus improving recommendation accuracy and interpretability.

Training and Recommendation RippleNet is trained by maximizing the posterior probability of model parameters Θ given the knowledge graph G and user-item interaction matrix Y, which is equivalent to maximizing

$$p(\Theta|G, Y) \propto p(\Theta) \cdot p(G|\Theta) \cdot p(Y|\Theta, G), \tag{3.30}$$

where $p(\Theta)$ is a Gaussian prior over embeddings, $p(G|\Theta)$ is the likelihood of the observed knowledge graph based on tensor factorization, and $p(Y|\Theta, G)$ is the likelihood of observed implicit feedback modeled by Bernoulli distributions.

Taking the negative log, the resulting loss function combines three terms: cross-entropy loss for user-item interactions, squared error for knowledge graph reconstruction, and L_2 regularization for embed-

dings and relations:

$$\mathcal{L} = \sum_{(u,v)\in Y} -\big(y_{uv}\log\sigma(\mathbf{u}^\top\mathbf{v}) + (1-y_{uv})\log(1-\sigma(\mathbf{u}^\top\mathbf{v}))\big) + \frac{\lambda_2}{2}\sum_{r\in R}\|I_r - E^\top R E\|_2^2 + \frac{\lambda_1}{2}(\|V\|_2^2 + \|E\|_2^2 + \sum_{r\in R}\|R\|_2^2). \tag{3.31}$$

Optimization is performed using stochastic gradient descent (SGD) on minibatches of interactions and triples sampled from Y and G, with parameter updates via back-propagation. After training, user and item embeddings are used to predict click probabilities for recommendation.

3.4.4. Knowledge Graph Attention Based Methods

Knowledge Graph Attention Network (KGAT) is a recommendation model that explicitly models high-order connectivities in a Collaborative Knowledge Graph (CKG), which consists of the user–item interaction graph and an auxiliary knowledge graph. KGAT addresses the limitations of traditional collaborative filtering (CF) and supervised feature-based methods by integrating structured side information from KGs and capturing long-range relational signals in an end-to-end fashion.

Embedding Layer KGAT begins by learning embedding representations for entities and relations in the KG. It applies a knowledge graph embedding method such as TransR[10] to project entities into relation-specific spaces. Given a triple (h, r, t), this projection relation can be written as:

$$\mathbf{e}_h^r + \mathbf{e}_r \approx \mathbf{e}_t^r, \tag{3.32}$$

where $\mathbf{e}_h, \mathbf{e}_t \in \mathbb{R}^d$ are the embeddings of head and tail entities, $\mathbf{e}_r \in \mathbb{R}^k$ is the relation embedding, and $\mathbf{e}^r = W_r\mathbf{e}$ denotes the projected representation in the relation space. The plausibility score of a triple is computed as:

$$g(h, r, t) = \|W_r\mathbf{e}_h + \mathbf{e}_r - W_r\mathbf{e}_t\|_2^2, \tag{3.33}$$

where $W_r \in \mathbb{R}^{k\times d}$ is a learnable transformation matrix for relation r. Lower scores indicate more plausible triples.

To optimize the knowledge graph embedding, KGAT adopts a pairwise ranking loss that encourages valid triples to obtain lower scores than corrupted ones. Specifically, for each observed triple (h, r, t), a negative (broken) triple (h, r, t') is constructed by randomly replacing the tail entity. The pairwise ranking loss is defined as:

$$\mathcal{L}_{KG} = \sum_{(h,r,t,t')\in\mathcal{T}} -\ln\sigma\big(g(h, r, t') - g(h, r, t)\big), \tag{3.34}$$

where

$$\mathcal{T} = \{(h, r, t, t') \mid (h, r, t) \in G,\ (h, r, t') \notin G\}, \tag{3.35}$$

and (h, r, t') denotes a corrupted triple constructed by randomly replacing one entity in a valid triple. $\sigma(\cdot)$ is the sigmoid function.

This objective enforces that the score of a valid triple $g(h, r, t)$ is smaller than that of a corrupted triple $g(h, r, t')$, thereby improving the discriminative capability of the knowledge graph embedding.

Attentive Embedding Propagation To capture high-order connectivity, KGAT recursively propagates embeddings along a node's neighbors using an attention mechanism. For an entity h, let $\mathcal{N}_h = \{(h, r, t) \mid (h, r, t) \in G\}$ denote its ego-network[19]. The aggregated neighbor representation is:

$$\mathbf{e}_{\mathcal{N}_h} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)\,\mathbf{e}_t, \tag{3.36}$$

where $\pi(h,r,t)$ is the attention weight for the neighbor t under relation r. The attention score is defined as:

$$\pi(h,r,t) = \frac{\exp\big((W_r\mathbf{e}_t)^\top \tanh(W_r\mathbf{e}_h + \mathbf{e}_r)\big)}{\sum_{(h,r',t')\in\mathcal{N}_h} \exp\big((W_{r'}\mathbf{e}_{t'})^\top \tanh(W_{r'}\mathbf{e}_h + \mathbf{e}_{r'})\big)}, \tag{3.37}$$

which captures the semantic importance of neighbor t relative to h and relation r. This attention mechanism allows KGAT to differentiate the contribution of neighbors during propagation, leading to more informative node representations.

Information Aggregation Once neighbor information is aggregated, the updated representation of node h is obtained by combining its original embedding with its aggregated neighbor embedding. One commonly used aggregator in KGAT is the GCN-based[8] update:

$$\mathbf{e}_h^{(l)} = \mathrm{LeakyReLU}\big(W(\mathbf{e}_h^{(l-1)} + \mathbf{e}_{\mathcal{N}_h}^{(l-1)})\big), \tag{3.38}$$

where l denotes the propagation layer. Other aggregators such as GraphSAGE[6] can also be used to model different types of interactions between a node and its neighbors.

Training and Recommendation After L layers of propagation, user and item representations are obtained as:

$$\mathbf{e}_u^* = \mathbf{e}_u^{(0)}\|\mathbf{e}_u^{(1)}\|\cdots\|\mathbf{e}_u^{(L)}, \quad \mathbf{e}_v^* = \mathbf{e}_v^{(0)}\|\mathbf{e}_v^{(1)}\|\cdots\|\mathbf{e}_v^{(L)}, \tag{3.39}$$

where $\|$ denotes concatenation across layers. The matching score between user u and item v is predicted by:

$$\hat{y}(u,v) = (\mathbf{e}_u^*)^\top \mathbf{e}_v^*. \tag{3.40}$$

KGAT is optimized using a combination of the knowledge graph embedding loss and a ranking loss such as Bayesian Personalized Ranking (BPR)[22]:

$$\mathcal{L}_{CF} = \sum_{(u,i,j)\in O} -\ln\sigma\big(\hat{y}(u,i) - \hat{y}(u,j)\big), \tag{3.41}$$

$$\mathcal{L}_{KGAT} = \mathcal{L}_{KG} + \mathcal{L}_{CF} + \lambda\|\Theta\|_2^2, \tag{3.42}$$

where O represents the set of observed positive and sampled negative interactions, and Θ denotes all model parameters.

By explicitly modeling the high-order relation structure in the graph and learning attentive propagation weights, KGAT is able to leverage rich semantic signals from the knowledge graph to improve recommendation accuracy and interpretability.

4

Results

4.1. Recommendation Performance

Table 4.1: Recommendation performance comparison of different models on various datasets. The results are reported in percentage (%) and best results for each metric are underlined.

Dataset	Model	NDCG@10	Recall@10	Hit Ratio@10	Precision@10
Beauty	PGPR	5.561	8.530	14.697	1.737
	TPRec	5.736	8.886	15.329	1.878
	TMER	11.413	9.951	24.359	4.888
	RippleNet	2.100	3.679	4.928	0.541
	KGAT	2.608	2.592	5.133	0.561
Cellphones	PGPR	3.782	5.945	8.337	0.862
	TPRec	5.682	9.548	13.565	1.457
	TMER	10.982	11.436	21.784	3.912
	RippleNet	1.159	2.110	5.058	0.519
	KGAT	1.901	2.669	3.827	0.395
Clothing	PGPR	2.871	4.830	7.006	0.726
	TPRec	3.106	5.297	7.699	0.803
	TMER	6.874	6.391	13.528	2.941
	RippleNet	1.242	1.391	4.046	0.105
	KGAT	1.684	2.103	3.612	0.412

4.1.1. Overall Recommendation Performance Comparison

From Table 4.1, it can be observed that **TMER** consistently achieves the best performance across all datasets and evaluation metrics. The improvements are particularly significant in terms of NDCG@10 and Precision@10, indicating that TMER not only retrieves more relevant items but also ranks them more accurately at top positions.

The superior performance of TMER can be attributed to three main factors. First, temporal interaction sequences are explicitly modeled, enabling the dynamic evolution of user preferences to be captured. In contrast, models such as RippleNet and KGAT rely primarily on static knowledge graph propagation or neighbor aggregation without modeling temporal dependencies. Second, meta-path guided reasoning is leveraged to extract structured semantic relationships between users and items. This structured

reasoning mechanism provides richer contextual signals compared to simple multi-hop propagation. Third, an attention-based aggregation mechanism is employed to focus on the most informative paths, thereby reducing noise from irrelevant connections and improving ranking precision.

Among the baselines, TPRec achieves the second-best performance in most cases. This can be attributed to its incorporation of time-aware path reasoning, which alleviates the limitation of purely static knowledge graph modeling. However, temporal information in TPRec is mainly injected during path construction, whereas temporal dynamics in TMER are more deeply integrated into sequential representation learning. This difference likely contributes to the stronger performance gains achieved by TMER.

PGPR shows moderate performance. Although reinforcement learning is adopted to explore explainable reasoning paths, the large search space of knowledge graphs and the difficulty of stable policy learning may restrict the model's ability to consistently identify optimal paths for accurate recommendation. Consequently, its recall and precision remain lower than those of TMER and TPRec.

RippleNet and KGAT perform relatively worse across all datasets. In RippleNet, user preferences are propagated over the knowledge graph in a ripple-like manner, which captures high-order connectivity but lacks explicit reasoning structure and temporal modeling. KGAT applies graph attention networks to learn higher-order neighbor representations; however, without explicit path reasoning or temporal sequence modeling, its capacity to capture fine-grained user intent is limited. These limitations explain their comparatively lower effectiveness in both ranking quality and hit rate.

4.1.2. Summary of Findings

Overall, the experimental results indicate that combining temporal dynamics with structured path reasoning and attention-based aggregation leads to significantly better recommendation performance. Models that neglect temporal evolution or rely solely on neighborhood aggregation are less effective in capturing complex user preference patterns. The consistent superiority of TMER across multiple datasets further highlights the importance of jointly modeling temporal information and semantic paths in knowledge graph-based recommender systems.

4.2. Explanation Performance

Table 4.2: Explaination performance of different models on multiple datasets across various metrics.

Dataset	Model	Temporal / Popularity / Diversity			Explanation-Ground Truth Consistency			Faithfulness	
		LIR	SEP	ETD	Prec	F1	Recall	JSf	JSw
Beauty	PGPR	0.1785	0.2254	0.1275	0.5432	0.1543	0.2705	0.0133	0.0119
	TPRec	0.2288	0.1893	0.1816	0.7068	0.2135	0.3243	0.2014	0.1779
	TMER	0.8532	0.1469	0.5771	0.0135	0.0019	0.0221	0.0148	0.0762
	RippleNet	0.1913	0.1514	0.2134	0.3743	0.0455	0.0502	0.3536	0.2532
	KGAT	0.1124	0.1687	0.2841	0.4265	0.1118	0.1896	0.2584	0.2232
Cellphones	PGPR	0.3552	0.1761	0.3154	0.1919	0.0505	0.1843	0.1198	0.1105
	TPRec	0.1782	0.2359	0.1978	0.4833	0.1581	0.3367	0.1873	0.1636
	TMER	0.8435	0.0002	0.2055	0.0148	0.0023	0.0219	0.0155	0.0843
	RippleNet	0.1078	0.1178	0.1971	0.3878	0.0591	0.0657	0.3654	0.2774
	KGAT	0.1721	0.1496	0.2618	0.3412	0.0927	0.1613	0.2419	0.2096
Clothing	PGPR	0.1929	0.1997	0.1155	0.5657	0.2092	0.3792	0.0360	0.0278
	TPRec	0.1983	0.1597	0.1021	0.6845	0.2627	0.4269	0.2318	0.2075
	TMER	0.8741	0.0006	0.2082	0.0120	0.0013	0.0127	0.0142	0.0690
	RippleNet	0.1595	0.1751	0.1907	0.4687	0.0536	0.0595	0.3931	0.3744
	KGAT	0.1413	0.1624	0.2479	0.4126	0.1089	0.1824	0.2687	0.2369

4.2.1. Overall Explanation Performance Comparison

Tables 4.2-4.4 present explanation performance from three complementary dimensions: (1) temporal, popularity, and diversity characteristics; (2) explanation-ground truth consistency; and (3) faithfulness. In addition to reporting overall performance on the Top-K recommended items, the evaluation is further decomposed into two additional subsets, namely relevant items and irrelevant items. Specifically, Tables 4.3 and 4.4 separately report explanation metrics for items that are correctly recommended (i.e., relevant) and those that are incorrectly recommended (i.e., irrelevant).

The inclusion of these additional tables is intended to disentangle explanation quality from recommendation correctness and, more importantly, to identify explanation metrics that can effectively reflect the quality of model explanations. By explicitly separating the results into relevant and irrelevant subsets, it becomes possible to evaluate whether an explanation metric is sensitive to the correctness of recommendations.

A reliable explanation metric is expected to exhibit clear differences between these two subsets. Specifically, for relevant items, higher scores should indicate that the generated explanations are well aligned with user preferences and underlying reasoning patterns. In contrast, for irrelevant items, explanation quality should degrade, reflecting the model's inability to provide meaningful or valid reasoning for incorrect predictions. Therefore, a significant performance gap between relevant and irrelevant subsets suggests that the metric is capable of distinguishing high-quality explanations from low-quality ones.

Conversely, if an explanation metric remains relatively stable across the two subsets, it suggests that the generated explanations are highly coupled with the model's underlying reasoning mechanism, and fail to effectively distinguish between correct and incorrect recommendations. In such cases, the explanations may merely reflect the internal computation of the model in a mechanical manner, rather than capturing the validity or meaningfulness of the recom-

Table 4.3: Explanation performance of different models on relevant items across various metrics.

Dataset	Model	Temporal / Popularity / Diversity			Explanation-Ground Truth Consistency			Faithfulness	
		LIR	SEP	ETD	Prec	F1	Recall	JSf	JSw
Beauty	PGPR	0.0675	0.2403	0.5909	0.2314	0.1235	0.1827	0.0105	0.0092
	TPRec	0.0756	0.2121	0.6009	0.4186	0.1714	0.2368	0.1846	0.1623
	TMER	0.8798	0.1255	0.7483	0.0327	0.0031	0.0125	0.0129	0.0708
	RippleNet	0.2150	0.1702	0.2984	0.2861	0.0408	0.0446	0.3320	0.2395
	KGAT	0.2214	0.2098	0.2986	0.3427	0.1215	0.2879	0.2508	0.2164
Cellphones	PGPR	0.0723	0.2126	0.8353	0.0925	0.0550	0.0284	0.0108	0.0096
	TPRec	0.0807	0.2330	0.5996	0.2914	0.1216	0.2148	0.1712	0.1498
	TMER	0.8300	0.0001	0.1900	0.0300	0.0040	0.0180	0.0138	0.0750
	RippleNet	0.1272	0.1400	0.2667	0.2910	0.0476	0.0532	0.3410	0.2608
	KGAT	0.2057	0.1936	0.2742	0.3268	0.1107	0.2684	0.2389	0.2013
Clothing	PGPR	0.0323	0.2262	0.7000	0.3048	0.1467	0.2185	0.0314	0.0249
	TPRec	0.0317	0.1815	0.6789	0.4528	0.2114	0.3186	0.2146	0.1913
	TMER	0.8600	0.0005	0.1950	0.0250	0.0030	0.0100	0.0128	0.0600
	RippleNet	0.1768	0.1850	0.2521	0.3625	0.0462	0.0508	0.3658	0.3471
	KGAT	0.2169	0.2017	0.2894	0.3346	0.1189	0.2987	0.2495	0.2128

mendation outcomes. As a result, such metrics may lack discriminative power in evaluating explanation quality and may not reliably measure the true value of the explanations.

Temporal, Popularity, and Diversity When analyzing the explanation behavior of different models, an important observation emerges: *models with similar underlying mechanisms tend to exhibit similar explanation patterns*. Therefore, instead of following conventional categorizations (e.g., path-based, embedding-based, or hybrid-based), the models based on their learning and reasoning mechanisms are reorganized. Specifically, they are classified into three categories: (1) reinforcement learning-based path exploration methods (PGPR and TPRec), (2) explicit path reasoning methods (TMER), and (3) embedding propagation-based models (RippleNet and KGAT). This perspective enables a clearer understanding of how different modeling strategies influence explanation characteristics.

From the perspective of temporal alignment (LIR), a clear and consistent pattern can be observed across all datasets. TMER achieves significantly higher LIR scores than all other models (e.g., 0.8493 on Beauty and 0.8900 on Clothing), indicating that its explicitly constructed temporally-aware reasoning paths effectively capture users' recent preferences. By incorporating temporal ordering into the reasoning process, TMER ensures that entities in the explanation paths are closely aligned with users' latest interactions.

In comparison, reinforcement learning-based models (TPRec and PGPR) exhibit moderate LIR values. For instance, TPRec reaches 0.2872 on Beauty and 0.2544 on Clothing, while PGPR ranges approximately from 0.21 to 0.38 across datasets. This suggests that temporal signals can be partially incorporated through policy optimization, but the lack of explicit structural constraints limits their ability to fully capture recency patterns. Embedding propagation-based models (RippleNet and KGAT) show the lowest LIR values overall (e.g., RippleNet achieves only 0.0711 on Clothing), indicating that propagation mechanisms are less effective in modeling temporal dynamics.

Table 4.4: Explanation performance of different models on irrelevant items across various metrics.

Dataset	Model	Temporal / Popularity / Diversity			Explanation-Ground Truth Consistency			Faithfulness	
		LIR	SEP	ETD	Prec	F1	Recall	JSf	JSw
Beauty	PGPR	0.2164	0.2131	0.1008	0.0458	0.0270	0.0369	0.0158	0.0146
	TPRec	0.2872	0.1879	0.1652	0.0957	0.0463	0.0712	0.2198	0.1965
	TMER	0.8493	0.2159	0.5342	0.0433	0.0054	0.0235	0.0176	0.0839
	RippleNet	0.1881	0.1511	0.2095	0.1037	0.0165	0.0238	0.3742	0.2730
	KGAT	0.1086	0.1564	0.2517	0.1028	0.0346	0.0523	0.2717	0.2289
Cellphones	PGPR	0.3827	0.1518	0.1772	0.0378	0.0200	0.0152	0.0157	0.0142
	TPRec	0.1970	0.2425	0.1445	0.0725	0.0354	0.0589	0.2045	0.1817
	TMER	0.8600	0.0005	0.2200	0.0420	0.0060	0.0250	0.0175	0.0910
	RippleNet	0.0957	0.1156	0.1893	0.0958	0.0178	0.0239	0.3895	0.2968
	KGAT	0.1594	0.1473	0.2136	0.0941	0.0308	0.0447	0.2826	0.2041
Clothing	PGPR	0.2187	0.2073	0.1451	0.0826	0.0415	0.0673	0.0407	0.0336
	TPRec	0.2544	0.1499	0.1016	0.1183	0.0649	0.0927	0.2499	0.2248
	TMER	0.8900	0.0012	0.2205	0.0380	0.0050	0.0150	0.0165	0.0750
	RippleNet	0.0711	0.0583	0.1822	0.0954	0.0148	0.0207	0.4189	0.3996
	KGAT	0.1328	0.1545	0.2198	0.1076	0.0392	0.0614	0.2773	0.2447

Overall, these results demonstrate that explicit temporal modeling significantly improves the alignment between explanations and users' recent preferences compared to implicit approaches.

Regarding popularity bias (SEP), lower values indicate reduced reliance on globally popular entities. TMER consistently achieves the lowest or near-lowest SEP values across all datasets, in some cases approaching zero (e.g., 0.0005 on Cellphones and 0.0012 on Clothing). This suggests that its constrained path reasoning mechanism effectively mitigates the overemphasis on high-degree entities. In contrast, reinforcement learning-based models (PGPR and TPRec) exhibit relatively higher SEP values, indicating that frequently occurring entities still influence the exploration process. RippleNet and KGAT demonstrate intermediate SEP values, reflecting the inherent tendency of embedding propagation to amplify high-degree nodes.

These findings indicate that explicit reasoning constraints are more effective in controlling popularity bias, while embedding-based propagation is more susceptible to the structural properties of the knowledge graph.

In terms of diversity (ETD), TMER achieves the highest score on the Beauty dataset, demonstrating that its meta-path guided reasoning can effectively explore structurally diverse semantic relations. However, on Cellphones and Clothing, its ETD values are comparable to other models, suggesting that its diversity advantage may depend on dataset characteristics. Reinforcement learning-based models (PGPR and TPRec) show moderate diversity (e.g., TPRec achieves 0.1652 on Beauty), reflecting their ability to explore multiple candidate paths. Embedding propagation-based models (RippleNet and KGAT) generally exhibit lower or moderate diversity, although KGAT occasionally achieves competitive ETD values, likely due to its attention-based aggregation over multiple neighbors.

Overall, a consistent trend emerges: explicit reasoning models (e.g., TMER) excel in temporal alignment and popularity control; reinforcement learning-based models strike a balance between flexibility and structural guidance; and

embedding propagation-based models are more influenced by the structural characteristics of the knowledge graph.

Explanation-Ground Truth Consistency A different trend emerges when examining explanation–ground truth consistency metrics (Precision, Recall, and F1). Embedding-based models with reinforcement learning, particularly TPRec, achieve the highest Precision and F1 scores across most datasets. PGPR follows closely. This indicates that reinforcement learning-based path selection tends to generate explanation entities that exhibit stronger lexical overlap with review-derived rationales.

TPRec benefits from a time-aware path filtering mechanism incorporated during reinforcement learning. This mechanism enables the model to be more sensitive to users' recent preference dynamics, prioritizing items that are more likely to match the user's interests. As a result, the reasoning paths not only become semantically more meaningful but also better align with the user's behavioral intentions. PGPR, which also employs reinforcement learning for path exploration, exhibits slightly lower consistency compared to TPRec due to its larger search space and the inherent instability in its exploration process.

KGAT achieves moderate consistency scores. Although graph attention mechanisms capture higher-order structural relevance, explanations are derived indirectly from embedding weights rather than explicit reasoning paths, limiting semantic interpretability. RippleNet shows similar moderate-to-low consistency performance, reflecting its reliance on preference propagation rather than structured reasoning.

In contrast, TMER demonstrates relatively lower lexical-level Precision, Recall, and F1 scores. This is because, by design, TMER explicitly defines the types of reasoning paths it generates (details in Section 3.4.2), and these paths do not include entities of type "Word." Consequently, when computing Precision, Recall, and F1, only entities of type "brand" and "category" that appear in the explanation paths are considered as the model's predicted explanations (S_u). This restricted ground truth coverage naturally leads to substantially lower values, even though temporally guided path reasoning effectively improves recommendation accuracy and recency alignment.

Faithfulness Faithfulness is measured via JS divergence between the model's internal reasoning distribution and the distribution implied by generated explanations. Lower values indicate stronger alignment.

PGPR achieves particularly low JS divergence across datasets. Since reinforcement learning explicitly optimizes path policies, explanation paths closely reflect the model's internal decision-making distribution. TPRec also demonstrates relatively low divergence, benefiting from structured policy constraints and temporal filtering.

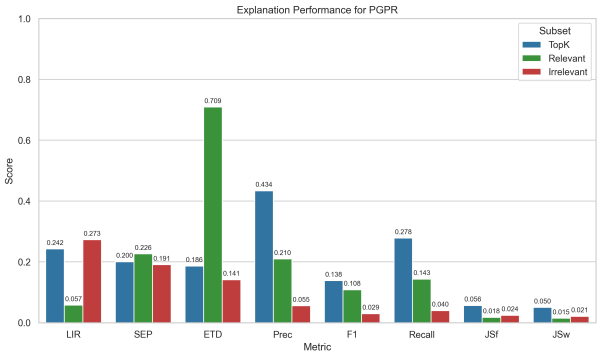
TMER achieves moderate-to-low JS values. Although attention-based aggregation introduces flexibility in path weighting, explanation generation remains grounded in explicit meta-path reasoning, preserving a certain degree of faithfulness.

RippleNet and KGAT generally present higher JS divergence. Embedding propagation and attention weighting distribute importance across numerous neighbors without discrete reasoning trajectories. Consequently, extracted explanations may not precisely reflect stable internal reasoning structures.

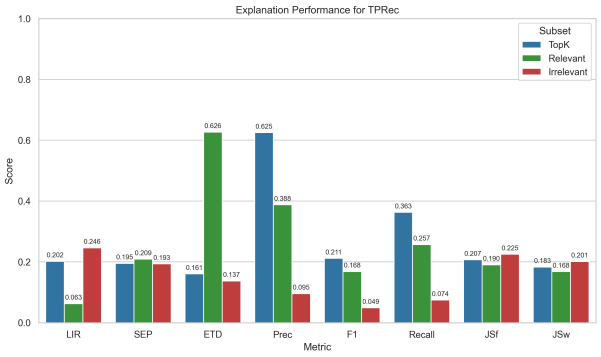
4.2.2. Relevant vs. Irrelevant Items

Figure 4.1 shows a comparison of Top-K, relevant, and irrelevant items across five recommendation models (PGPR, TPRec, TMER, RippleNet, and KGAT). Each value in the visualization represents the average across the three datasets (Beauty, Cellphones, and Clothing), which makes it easy to intuitively assess each model's ability to rank relevant items higher in the Top-K recommendations and provides a clear understanding of performance differences between models.

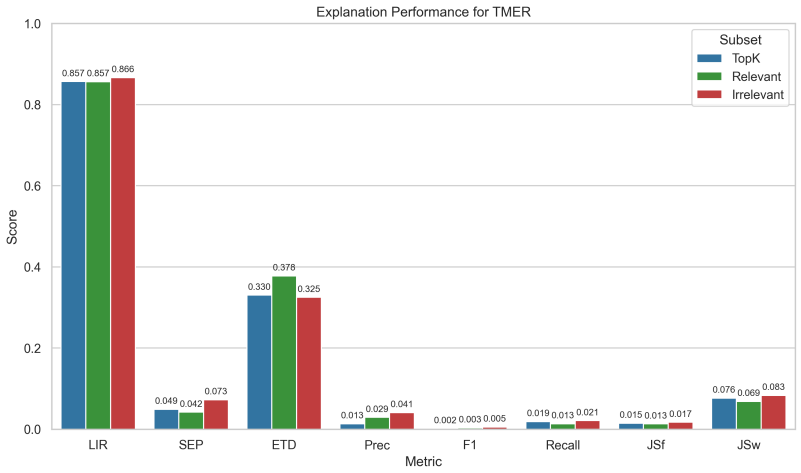
Temporal, Popularity, and Diversity When analyzing LIR, SEP, and ETD, distinct patterns emerge across models and item subsets.



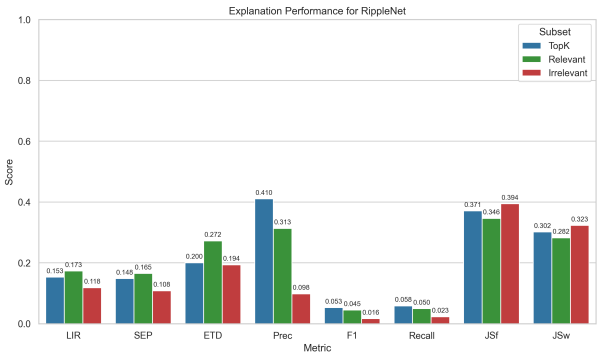
(a) PGPR



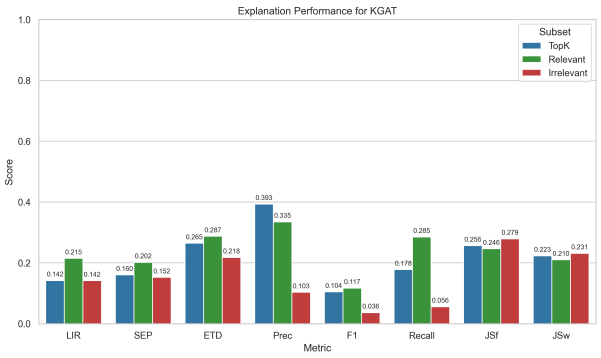
(b) TPRec



(c) TMER



(d) RippleNet



(e) KGAT

Figure 4.1: Comparison of Top-K, relevant, and irrelevant items across different models.

For TMER as well as PGPR and TPRec, LIR is typically higher on irrelevant items than on Top-K or relevant items. This indicates that these models rely heavily on recent interactions even when recommendations are incorrect, reflecting their intrinsic temporal sensitivity. However, high LIR alone does not guarantee correct recommendations, suggesting that temporal proximity is necessary but insufficient for accuracy.

In contrast, RippleNet and KGAT show higher LIR on relevant items, indicating that their temporal reasoning is better aligned with recommendation correctness. For these models, explanations grounded in recent interactions are more likely to correspond to successful recommendations.

Regarding SEP, relevant items generally exhibit the highest values across almost all models. This implies that correct and high-ranked recommendations rely more on globally popular entities, highlighting that widely connected or frequently occurring entities not only aid recommendation outcomes but also enhance explanation relevance. For TMER, PGPR, and TPRec, SEP for irrelevant items sometimes slightly increases, suggesting that incorrect recommendations may also rely on popular entities, but this does not guarantee accuracy.

With respect to ETD, relevant items usually have higher diversity than irrelevant items, particularly for PGPR, TPRec, RippleNet, and KGAT. This indicates that successful recommendations are supported by a richer variety of reasoning patterns, improving interpretability and semantic richness. For TMER, ETD shows a similar trend.

Overall, these observations reveal the relationship between explanation characteristics and recommendation correctness: higher LIR on irrelevant items shows over-reliance on recent interactions without ensuring accuracy, whereas higher SEP and ETD on relevant items indicate that correct recommendations depend on globally informative entities and diverse reasoning patterns.

Explanation-Ground Truth Consistency For embedding-based models with reinforcement learning, particularly TPRec and PGPR, explanation-ground truth consistency metrics (Precision, Recall, and F1) exhibit a clear performance gap between relevant and irrelevant items. Across all datasets, these models achieve substantially higher Precision and F1 scores on relevant items than on irrelevant ones. For example, TPRec consistently maintains strong Precision and Recall on relevant items, while these metrics drop noticeably for irrelevant items. This indicates that their explanation generation is tightly coupled with successful ranking outcomes: when the recommendation is correct, the reasoning paths tend to align well with user review rationales; when the ranking fails, explanation quality deteriorates.

Interestingly, the aggregated metrics computed over the entire Top-K recommended items sometimes appear higher than the metrics computed only over relevant items. This phenomenon can be explained by the definitions of Precision and Recall given in Equation 3.1.

Since Top-K includes both relevant and irrelevant items, the union of their explanation paths S_u can contain additional overlaps with ground-truth entities G_u. Although irrelevant items are not correctly recommended, some of their explanation paths may still partially match user review entities. When averaging over all Top-K items, this partial overlap can increase the numerator of Precision and Recall, while the $+1$ in the denominators reduces sensitivity to the total set size. As a result, Top-K aggregated metrics can occasionally exceed those calculated solely on the relevant subset, without contradicting the expected trend that relevant items generally have higher quality explanations.

In contrast, KGAT and RippleNet show moderate but more uniform behavior. Although their overall consistency scores are lower than those of reinforcement learning-based models, the performance degradation from relevant to irrelevant subsets is relatively less dramatic.

TMER exhibits consistently low lexical-level Precision and F1 across both subsets. However, the difference between relevant and irrelevant items is relatively small. This stability stems from the predefined path-type constraints (as described in Section 3.4.2), where explanation paths exclude “Word”-type entities. Consequently, the ground-truth overlap is computed only over brand and category entities, limiting the attainable lexical overlap regardless of ranking

correctness. Therefore, TMER's explanation-ground truth consistency is structurally bounded rather than strongly influenced by recommendation success.

Faithfulness Faithfulness results (JS divergence) further highlight structural differences among models. Reinforcement learning-based models generally maintain relatively low JS divergence for both relevant and irrelevant items, indicating that generated explanations consistently reflect internal rule distributions. Although minor increases are observed for irrelevant items, the divergence remains comparatively stable, implying that explanation faithfulness is not entirely contingent on ranking success.

By contrast, KGAT and RippleNet show higher JS divergence overall, and the difference between relevant and irrelevant subsets is less pronounced. This suggests that their explanation extraction mechanisms are less tightly aligned with discrete reasoning rules and more dependent on distributed embedding weights, resulting in weaker structural faithfulness irrespective of correctness.

TMER maintains low-to-moderate JS divergence in both subsets, demonstrating that its explicit meta-path reasoning preserves alignment between generated paths and training rule distributions, even when recommendations are incorrect.

4.3. Trade-off Between Recommendation Accuracy and Explanation Quality

While previous sections independently evaluated recommendation accuracy and explanation quality, a deeper analysis reveals an inherent trade-off between these two objectives. Optimizing ranking performance does not necessarily guarantee semantically aligned or lexically consistent explanations, and conversely, improving explanation-ground-truth consistency may not always yield superior recommendation accuracy. This section synthesizes the empirical findings to analyze this trade-off from multiple perspectives.

Accuracy vs. Lexical Consistency From the recommendation results in Table 4.1, TMER consistently achieves the best performance across all datasets in terms of NDCG@10, Recall@10, Hit Ratio@10, and Precision@10. However, when examining explanation–ground-truth consistency (Precision, Recall, and F1), TMER obtains significantly lower lexical overlap scores compared to TPRec and PGPR. This discrepancy is primarily structural rather than performance-driven.

In contrast, TPRec and PGPR adopt reinforcement learning-based path exploration over a larger search space that includes word-level entities. This design increases the probability of lexical overlap with review-derived ground truth, thereby improving Precision and F1 metrics. However, despite stronger lexical consistency, their recommendation accuracy remains inferior to TMER.

This observation highlights a key trade-off:

- Constraining explanation structure (as in TMER) enhances temporal robustness and ranking performance but reduces surface-level lexical alignment.
- Expanding explanation search space (as in TPRec and PGPR) improves lexical overlap but introduces additional noise and instability that may limit ranking effectiveness.

Temporal Alignment vs. Ranking Correctness Across all datasets and subsets (Top-K, relevant, and irrelevant), TMER achieves the highest LIR scores, demonstrating strong temporal sensitivity. Interestingly, LIR is often even higher for irrelevant items than for relevant ones. This indicates that strong recency alignment alone does not guarantee correct recommendations.

This phenomenon reflects another trade-off dimension. Temporal signals are highly informative but insufficient in isolation. Excessive reliance on recent interactions may bias the model toward short-term preferences while overlooking

stable long-term interests or semantic compatibility. Thus, optimizing temporal alignment can improve interpretability and responsiveness but does not automatically maximize accuracy.

Embedding-based propagation models such as RippleNet and KGAT exhibit weaker temporal alignment but sometimes show better alignment between LIR and recommendation correctness. Their performance suggests that moderate temporal sensitivity combined with global structural aggregation may yield more balanced decision boundaries, albeit at the cost of overall ranking performance.

Popularity Bias and Diversity Considerations The SEP metric reveals that relevant items typically exhibit higher popularity bias across models. Correct recommendations often depend on well-connected or frequently occurring entities in the knowledge graph. This suggests that leveraging globally popular entities can enhance ranking accuracy.

However, excessive dependence on popularity may reduce explanation diversity and personalization. TMER maintains relatively low SEP values while preserving high ETD diversity scores. Its meta-path constraints encourage structurally diverse reasoning patterns, improving interpretability without excessively amplifying degree-based bias.

RippleNet and KGAT, which rely on embedding propagation, tend to inherit degree distribution characteristics of the knowledge graph. While this may stabilize training and provide moderate consistency, it can amplify popularity bias and reduce explanation diversity.

Therefore, another trade-off emerges:

- Utilizing high-degree entities improves ranking reliability.
- Encouraging diverse meta-path exploration enhances interpretability and semantic richness but may introduce additional variance into ranking optimization.

4.4. Analysis of Ripple Set Construction

This section investigates how the construction of the ripple set affects recommendation performance. Specifically, the impact of ripple set size and different neighbor sampling strategies is analyzed.

4.4.1. Impact of Ripple Set Size

RippleNet models user preferences by propagating user interests through a knowledge graph. It starts from the items that a user has previously interacted with and treats the corresponding entities as seed entities. These seed entities serve as the starting point for multi-hop exploration in the knowledge graph. At each hop, the model expands from the current set of entities by following knowledge graph relations to their neighboring entities.

The entities and relations collected at each hop form what is known as a *ripple set*. These ripple sets allow the model to propagate user preference signals layer by layer across the knowledge graph, enabling the recommendation model to incorporate structured knowledge when predicting user-item interactions.

During the multi-hop expansion process, the number of neighboring entities can grow rapidly, which leads to significant computational costs. To control the size of the ripple sets, RippleNet samples a fixed number of neighbors at each hop. In the original implementation, RippleNet adopts a *uniform sampling* strategy. Under this strategy, neighbors are randomly selected from all available neighbors of an entity with equal probability. This approach is simple and efficient, ensuring that the ripple set size remains manageable during multi-hop propagation.

However, uniform sampling does not consider the importance or relevance of neighbors, which may result in the inclusion of less informative or noisy entities. Motivated by this limitation, the effects of different ripple set sizes and alternative neighbor sampling strategies on recommendation performance are further investigated.

To analyze this effect, experiments are conducted with different ripple set sizes, including 32, 64, 128, and 256. For each configuration, the same ripple set size is applied at every hop while keeping all other hyperparameters

Table 4.5: Impact of Ripple Set Size on Recommendation Performance

Ripple Set Size	Beauty				Cellphones				Clothing			
	NDCG@10	Recall@10	HR@10	Precision@10	NDCG@10	Recall@10	HR@10	Precision@10	NDCG@10	Recall@10	HR@10	Precision@10
32	1.954	3.241	4.356	0.498	1.083	1.972	4.821	0.487	1.103	1.212	3.864	0.094
64	**2.100**	**3.679**	**4.928**	**0.541**	**1.159**	**2.110**	**5.058**	**0.519**	1.187	1.324	3.972	0.101
128	2.047	3.514	4.761	0.528	1.124	2.041	4.936	0.502	**1.242**	**1.391**	**4.046**	**0.105**
256	1.981	3.386	4.602	0.511	1.087	1.965	4.782	0.483	1.201	1.337	3.981	0.101

unchanged. The evaluation is performed on three datasets: Beauty, Cellphones, and Clothing. The results are reported in Table 4.5, where NDCG@10, Recall@10, HR@10, and Precision@10 are used as evaluation metrics.

From the results, it can be observed that the best performance on the Beauty and Cellphones datasets is achieved when the ripple set size is set to 64, while the Clothing dataset achieves slightly better results with a ripple set size of 128. Considering both recommendation performance, a ripple set size of 64 is adopted for the Beauty and Cellphones datasets, while 128 is adopted for the Clothing dataset as the default setting in the subsequent experiments.

4.4.2. Comparison of Neighbor Sampling Strategies

Let the ripple set of a user u be defined as

$$R_u = \{e_1, e_2, \ldots, e_N\},$$

where each e_i represents a neighboring entity in the knowledge graph. The goal is to sample a fixed number K of neighbors from R_u. Each entity e_i is associated with several attributes used to guide the sampling process:

- t_i: the most recent interaction time related to the entity;
- pop_i: the popularity of the entity (e.g., the frequency of occurrence in the dataset);
- $\mathbf{v}_i$: the embedding vector representing the semantic representation of the entity.

Instead of uniform sampling, several non-uniform sampling strategies guided by temporal recency, popularity, and diversity are designed.

Recency-Aware Sampling (LIR-guided Sampler) Since user interests evolve over time, recent interactions tend to better reflect current preferences. Entities with more recent interactions are assigned higher sampling probabilities. Entities in the ripple set are first sorted according to their interaction timestamps to obtain the ranking $rank(e_i)$, where $rank(e_i) = 1$ denotes the most recent entity and $rank(e_i) = N$ the oldest entity. For entities that do not correspond to items, such as brands, categories, or other abstract entities, direct interaction timestamps may be unavailable. In such cases, the timestamp of the entity is inherited from its associated source item in the ripple set. This allows recency-aware sampling to be applied consistently across all entity types, ensuring that even non-item entities are prioritized based on the temporal relevance of the user's interactions. The sampling probability is defined as

$$p_i = \frac{N - rank(e_i) + 1}{\sum_{j=1}^{N}(N - rank(e_j) + 1)}$$

Finally, K entities are sampled according to p_i. This strategy emphasizes recent interactions and captures short-term user interests.

Popularity-Regularized Sampling (SEP-guided Sampler) Entity popularity often reflects strong collaborative signals, but focusing only on highly popular entities may reduce long-tail coverage. Entities are ranked according to their popularity pop_i, producing a ranking $rank(e_i)$ where $rank(e_i) = 1$ corresponds to the most popular entity. Sampling

probabilities are assigned as

$$p_i = \frac{rank(e_i)}{\sum_{j=1}^{N} rank(e_j)}$$

A subset of K entities is then sampled based on p_i. This approach incorporates popularity information while allowing less popular entities to be selected.

Diversity-Promoting Sampling (ETD-guided Sampler) Selecting highly similar neighbors may lead to redundant information propagation. To enhance diversity, both embeddings and entity types are considered to promote selection of semantically and categorically distinct entities. Let S_{selected} denote the set of already selected entities. The first entity is chosen randomly. For each remaining entity e_i, a diversity weight is calculated by combining the maximum cosine similarity with already selected entities and the entity type coverage:

$$w_i = \frac{1 - \max_{e_j \in S_{\text{selected}}} \text{sim}(\mathbf{v}_i, \mathbf{v}_j) + \mathbf{1}\{c_i \notin \{c_j : e_j \in S_{\text{selected}}\}\}}{2},$$

where $\mathbf{v}_i$ is the embedding vector and c_i is the type of entity e_i. The sampling probability is then given by

$$p_i = \frac{w_i}{\sum_{e_k \in R_u \setminus S_{\text{selected}}} w_k}.$$

Entities are sampled iteratively according to p_i until K neighbors are selected. This strategy ensures both semantic and type diversity among the sampled neighbors while reducing redundancy during information propagation.

Table 4.6: Performance comparison between baseline and different non-uniform sampling strategies on three datasets. Accuracy metrics are reported in percentage (%).

Dataset	Sampling Strategy	Temporal / Popularity / Diversity			Recommendation Accuracy (%)			
		LIR	SEP	ETD	NDCG@10	Recall@10	HR@10	Prec@10
Beauty	Baseline	0.1913	0.1514	0.2134	2.103	3.681	4.931	0.493
	LIR-guided	0.2105	0.1497	0.2125	2.148	3.748	5.012	0.501
	SEP-guided	0.1872	0.1706	0.2055	2.096	3.642	4.865	0.487
	ETD-guided	0.1854	0.1521	0.2203	2.186	3.719	5.063	0.506
Cellphones	Baseline	0.1078	0.1178	0.1971	1.162	2.115	5.061	0.506
	LIR-guided	0.1105	0.1152	0.1938	1.178	2.164	5.124	0.512
	SEP-guided	0.1083	0.1262	0.1955	1.169	2.108	5.082	0.508
	ETD-guided	0.1057	0.1184	0.2106	1.192	2.118	5.163	0.516
Clothing	Baseline	0.1595	0.1751	0.1907	1.245	1.398	4.052	0.405
	LIR-guided	0.1672	0.1642	0.1873	1.271	1.442	4.103	0.410
	SEP-guided	0.1421	0.1864	0.1916	1.224	1.395	4.071	0.407
	ETD-guided	0.1558	0.1723	0.2049	1.241	1.418	4.131	0.413

Table 4.6 presents the performance of the baseline and different non-uniform sampling strategies across the three datasets. Both temporal/popularity/diversity metrics (LIR, SEP, ETD) and recommendation accuracy metrics (NDCG@10, Recall@10, HR@10, Precision@10) are reported.

The results indicate that the proposed sampling strategies behave as designed. Specifically, the LIR-guided sampler consistently increases the LIR score on all datasets, reflecting effective recency-aware neighbor selection. The SEP-guided sampler raises the SEP score by prioritizing popular neighbors. The ETD-guided sampler achieves the highest ETD score, demonstrating its ability to promote diversity in the ripple set. These observations confirm that non-uniform sampling is effective in shaping the ripple set according to the desired properties, namely recency, popularity, and diversity.

Regarding recommendation accuracy, the impact of non-uniform sampling is generally stable. Accuracy metrics, including NDCG@10, Recall@10, HR@10, and Precision@10, exhibit only minor variations across the different sampling strategies. This suggests that the introduction of non-uniform sampling does not harm overall recommendation performance, while still providing additional control over the structural properties of the ripple set.

In summary, both the size and composition of the ripple set play important roles in influencing recommendation performance. An appropriately chosen ripple set size balances computational efficiency and model effectiveness, with the optimal size varying across datasets. Furthermore, non-uniform sampling strategies can effectively shape the ripple set to emphasize recency, popularity, or diversity, without compromising overall recommendation accuracy.

5

Conclusion

5.1. Key Findings

RQ1: Differences in Explanation Characteristics Across KGRS Paradigms Analysis of temporal, popularity, and diversity characteristics reveals that TMER, the explicit path reasoning model, consistently achieves the highest temporal alignment (LIR) and best control over popularity bias (SEP), effectively capturing users' recent preferences while avoiding over-reliance on popular entities. Its diversity (ETD) is also notable, particularly on the Beauty dataset, indicating that its meta-path guided reasoning explores semantically rich and structurally diverse paths. Reinforcement learning-based models, PGPR and TPRec, show moderate temporal alignment and diversity but weaker popularity control, while embedding propagation-based models, RippleNet and KGAT, exhibit the lowest temporal alignment and explanations more influenced by the structural properties of the knowledge graph.

Regarding explanation-ground truth consistency, TPRec and PGPR achieve the highest lexical-level precision and F1 on relevant items, demonstrating that their reasoning paths align well with user review rationales when recommendations are correct. In contrast, TMER shows the lowest lexical-level consistency due to its predefined path-type constraints, even though its temporally guided reasoning improves recommendation recency. RippleNet and KGAT display moderate but more uniform consistency, reflecting limited semantic alignment.

Faithfulness results further highlight structural differences among paradigms. PGPR and TPRec maintain low JS divergence, indicating that their explanations closely reflect internal reasoning distributions. TMER achieves moderate-to-low JS divergence, preserving alignment through explicit meta-path reasoning. RippleNet and KGAT, however, show higher JS divergence, suggesting that their embedding propagation mechanisms result in explanations that are less faithful to the model's internal decision-making.

Overall, these findings demonstrate that explanation characteristics are strongly determined by the underlying reasoning mechanisms. TMER excels in temporal robustness and popularity control, reinforcement learning-based models provide strong semantic alignment and flexibility, and embedding propagation-based models are more influenced by graph structure and less faithful to internal reasoning.

RQ2: Differences Between Explanations for Relevant and Irrelevant Recommended Items For almost all models, the differences between relevant and irrelevant items are more pronounced in Explanation-Ground Truth Consistency metrics than in other evaluation measures. For reinforcement learning-based models such as TPRec and PGPR, explanation quality is notably higher for relevant items than for irrelevant ones. This indicates that when the recommendation is correct, the generated reasoning paths align much better with the rationales expressed in user reviews, whereas explanation quality deteriorates significantly when the recommendation is incorrect.

This also suggests that Explanation-Ground Truth Consistency metrics, particularly Precision, Recall, and F1, can reliably distinguish high-quality explanations from low-quality ones. Since these metrics consistently achieve higher values for relevant items and lower values for irrelevant items, they provide a strong signal of whether an explanation is semantically meaningful and well aligned with the actual reasons behind user preferences.

Temporal, popularity, and diversity metrics also reveal important relationships between explanation quality and recommendation correctness. For TMER, PGPR, and TPRec, LIR is often higher for irrelevant items than for relevant items, indicating that these models rely heavily on users' recent interactions even when the recommendation is wrong. Therefore, temporal proximity alone is not sufficient to guarantee recommendation accuracy. In contrast, RippleNet and KGAT usually achieve higher LIR on relevant items, suggesting that their temporal signals are more consistently aligned with successful recommendations.

For SEP and ETD, most models show higher popularity bias and greater explanation diversity for relevant items than for irrelevant items. This indicates that correct recommendations are often supported by more representative popular entities and a richer set of reasoning patterns. In particular, TPRec, PGPR, RippleNet, and KGAT typically achieve higher ETD values on relevant items, suggesting that more diverse reasoning paths contribute to both semantic richness and recommendation success.

Faithfulness results further highlight differences across explanation mechanisms. TPRec and PGPR maintain relatively low JS divergence for both relevant and irrelevant items, indicating that their generated explanations remain well aligned with the model's internal reasoning distribution even when recommendations fail. TMER also preserves low-to-moderate JS divergence, showing that its explicit meta-path reasoning supports structurally faithful explanations across both subsets.

RQ3: Impact of Neighborhood Sample Size and Explanation-Oriented Sampling on Recommendation Accuracy and Explanation Quality The size and composition of RippleNet's ripple set significantly influence both recommendation performance and explanation characteristics. Experiments show that an intermediate ripple set size (64 for Beauty and Cellphones, 128 for Clothing) achieves the best trade-off between computational efficiency and ranking effectiveness. Smaller ripple sets may not capture sufficient multi-hop user preferences, while overly large ripple sets introduce noise and redundancy, slightly reducing accuracy.

Regarding neighbor sampling strategies, non-uniform sampling effectively shapes the ripple set according to specific objectives. LIR-guided sampling increases temporal alignment by prioritizing recently interacted entities, SEP-guided sampling raises the influence of popular entities, and ETD-guided sampling enhances diversity by promoting semantically and categorically distinct neighbors. Each strategy successfully adjusts the ripple set properties without substantially harming overall recommendation metrics (NDCG@10, Recall@10, HR@10, Precision@10).

In terms of explanation quality, these sampling strategies reinforce the link between ripple set composition and explanation characteristics. LIR-guided sampling emphasizes recent interactions, strengthening the temporal relevance of reasoning paths. SEP-guided sampling ensures explanations are grounded in well-connected, informative entities, while ETD-guided sampling increases the variety of explanation paths, enhancing semantic richness.

Overall, these findings suggest that by carefully choosing ripple set size and adopting explanation-oriented, non-uniform sampling, RippleNet can improve explanation quality across temporal, popularity, and diversity dimensions without compromising recommendation accuracy. This has practical significance: even when recommendation performance remains largely unchanged, fine-tuning the model structure allows the generated explanations to better match user preferences.

5.2. Limitations

Despite a comprehensive analysis of different KGRS paradigms, several limitations remain. First, our experiments are limited to three subsets from the Amazon Review Data dataset, namely Beauty, Cellphones, and Clothing, which

may restrict the generalizability of the results to other domains or larger-scale knowledge graphs. Since all three subsets are derived from the same Amazon platform, they share similar interaction patterns and knowledge structures. Therefore, the findings may not directly generalize to other recommendation domains beyond e-commerce, where user behaviors, item characteristics, and knowledge graph structures may differ substantially.

Second, our evaluation of explanation quality primarily relies on proxy metrics such as Explanation-Ground Truth Consistency (Precision, Recall, F1), LIR, SEP, ETD, and JS divergence. While these metrics provide interpretable and quantitative insights, they may not fully capture subjective aspects of explanation usefulness or user satisfaction. For instance, some users might prefer explanations emphasizing niche or novel entities rather than popular ones, which our current metrics cannot directly assess.

Finally, the experiments on RippleNet concerning ripple set size and non-uniform sampling strategies only consider a limited set of designs. More sophisticated adaptive or personalized sampling methods could potentially improve both recommendation accuracy and explanation alignment, but these approaches are beyond the scope of the current study. Additionally, non-uniform sampling introduces extra computational cost compared to the original uniform sampling, which should be considered when scaling to larger datasets.

5.3. Future Work

Several promising directions can be pursued to address the limitations of the current study.

First, extending experiments to additional domains and larger-scale knowledge graphs would help validate the generalizability of our findings. For example, conducting experiments in movie recommendation, music recommendation, or social media content recommendation could reveal how the relationship between explanation characteristics and recommendation correctness varies across datasets with different sparsity levels, interaction dynamics, and entity distributions. Such studies would deepen our understanding of KGRS performance across diverse application scenarios.

Second, incorporating user-centered evaluation methods can provide richer insights into explanation quality. User studies or surveys could assess the perceived usefulness and satisfaction of generated explanations, capturing subjective preferences that proxy metrics may not fully reflect.

Third, further optimization of RippleNet's ripple set construction is a promising direction. More sophisticated adaptive or personalized non-uniform sampling strategies, informed by user interests, context, or dynamic preferences, may improve both recommendation accuracy and explanation alignment. Such approaches could generate explanations that better match user expectations while maintaining model effectiveness. Additionally, computational efficiency and real-time applicability merit further investigation. Non-uniform sampling introduces additional computational overhead compared to uniform sampling, which may pose challenges in large-scale or online recommendation scenarios. Future work could explore parallel or distributed inference, approximate nearest neighbor search, incremental graph update mechanisms, and caching strategies to reduce latency and computation cost.

5.4. Summary of Thesis

This thesis systematically investigates the relationship between recommendation performance and explanation quality in knowledge graph-based recommender systems (KGRS). We compared different KGRS paradigms, including embedding-based, path-based, and hybrid methods, analyzing their differences in temporal alignment, popularity, diversity, lexical consistency, and faithfulness of explanations.

The experimental results show that TMER excels in temporal robustness and popularity control, reinforcement learning-based models achieve stronger semantic alignment and lexical consistency, while embedding propagation models are more influenced by graph structure but exhibit lower faithfulness. Differences between explanations for relevant and irrelevant items indicate that Explanation-Ground Truth Consistency metrics (Precision, Recall, F1) can reliably reflect explanation quality. In RippleNet, carefully selecting ripple set size and using explanation-oriented, non-uniform

sampling can improve explanation quality across temporal, popularity, and diversity dimensions without sacrificing recommendation accuracy.

Based on these findings, this thesis highlights key principles for designing KGRS: integrating temporal information with structured reasoning paths enhances both accuracy and interpretability; explanation metrics can guide adaptive model and sampling strategies; and trade-offs among different objectives should be balanced according to the application context. The study also highlights inherent trade-offs. Constraining explanation structure enhances ranking performance and temporal alignment but reduces lexical overlap, whereas expanding the reasoning search space improves lexical consistency at the cost of stability and accuracy. Temporal signals alone do not guarantee correct recommendations, and over-reliance on popular entities can improve ranking reliability but may reduce diversity and personalization. Balancing these objectives is crucial for designing interpretable and effective KGRS.

Looking forward, future work can build upon these findings by applying KGRS to diverse domains, such as movies, music, or social media, to validate the generalizability of explanation-driven design. Incorporating user-centered evaluations can provide richer insights into perceived explanation usefulness, capturing preferences that proxy metrics may not fully reflect. Furthermore, developing adaptive, context-aware sampling strategies could enhance both recommendation accuracy and explanation quality. Additional directions include improving computational efficiency for large-scale use.

5.5. Broader Implications and Responsible Deployment of Explainable KGRS

Overall, the findings contribute to a broader shift in the field from black-box accuracy maximization toward transparent, trustworthy, and controllable A I systems. Within this trajectory, the work empirically demonstrates that different reasoning mechanisms inherently encode different interpretability trade-offs. Moreover, the systematic comparison of temporal alignment, popularity bias, diversity, lexical consistency, and faithfulness across paradigms responds to the growing recognition that single-metric evaluation is insufficient for real-world deployment. This mirrors broader CS trends toward multi-objective optimization and responsible AI, where fairness, robustness, and explainability are treated as first-class citizens alongside predictive performance. Ultimately, the most important implication is that explainability must be designed in, not bolted on, and that designing for explainability requires making explicit trade-offs among temporal relevance, popularity control, diversity, lexical plausibility, and faithfulness. There is no universally best explanation; there are only context-appropriate choices.

However, as these design principles and trade-offs move from research to real-world deployment, they must confront a range of practical challenges and potential risks. Beyond technical limitations, deploying KGRS in real-world applications carries potential risks. For users, over-reliance on popularity-based explanations may reinforce filter bubbles and reduce exposure to diverse or niche content, potentially limiting serendipity and long-term satisfaction. For platform operators, explanations that are temporally aligned but lexically inconsistent could mislead users into overtrusting inaccurate recommendations, eroding trust over time. For content providers (e.g., sellers or creators), popularity-biased reasoning paths may disadvantage new or less-established items, raising fairness concerns. Addressing these risks requires not only better model design but also context-aware deployment strategies, user control over explanation preferences, and ongoing monitoring of unintended societal impacts.

6

Declaration of AI Use

Some parts content of this paper (mainly Introduction, Related work and Conclusion), the author used generative AI tools to enhance the quality and clarity of the academic writing. Specifically, these tools assisted in improving language fluency and refining the academic style, including better word choice and more precise expression. All generated contents were carefully reviewed by the author to ensure that no factual or conceptual deviation from the original research content occurred.

References

[1] Giacomo Balloccu et al. “Post Processing Recommender Systems with Knowledge Graphs for Recency, Popularity, and Diversity of Explanations”. In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’22. ACM, July 2022, pp. 646–656. DOI: 10.1145/3477495.3532041. URL: http://dx.doi.org/10.1145/3477495.3532041.

[2] Yixin Cao et al. “Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences”. In: *The World Wide Web Conference*. WWW ’19. ACM, May 2019, pp. 151–161. DOI: 10.1145/3308558.3313705. URL: http://dx.doi.org/10.1145/3308558.3313705.

[3] Hongxu Chen et al. *Temporal Meta-path Guided Explainable Recommendation*. 2021. arXiv: 2101.01433 [cs.SI]. URL: https://arxiv.org/abs/2101.01433.

[4] Rajarshi Das et al. *Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning*. 2018. arXiv: 1711.05851 [cs.CL]. URL: https://arxiv.org/abs/1711.05851.

[5] Luis Galárraga et al. “Fast rule mining in ontological knowledge bases with AMIE++”. In: *The VLDB Journal* 24.6 (Dec. 2015), pp. 707–730. ISSN: 1066-8888. DOI: 10.1007/s00778-015-0394-1. URL: https://doi.org/10.1007/s00778-015-0394-1.

[6] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs*. 2018. arXiv: 1706.02216 [cs.SI]. URL: https://arxiv.org/abs/1706.02216.

[7] Xiangnan He et al. *LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation*. 2020. arXiv: 2002.02126 [cs.IR]. URL: https://arxiv.org/abs/2002.02126.

[8] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG]. URL: https://arxiv.org/abs/1609.02907.

[9] Lovász László. “Random Walks on Graphs: A Survey, Combinatorics, Paul Erdos is Eighty”. In: *Bolyai Soc. Math. Stud.* 2 (Jan. 1993).

[10] Yankai Lin et al. “Learning entity and relation embeddings for knowledge graph completion”. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI’15. Austin, Texas: AAAI Press, 2015, pp. 2181–2187. ISBN: 0262511290.

[11] Qiang Liu et al. *Context-aware Sequential Recommendation*. 2016. arXiv: 1609.05787 [cs.IR]. URL: https://arxiv.org/abs/1609.05787.

[12] Scott Lundberg and Su-In Lee. *A Unified Approach to Interpreting Model Predictions*. 2017. arXiv: 1705.07874 [cs.AI]. URL: https://arxiv.org/abs/1705.07874.

[13] Thanet Markchom, Huizhi Liang, and James Ferryman. *Review of Explainable Graph-Based Recommender Systems*. 2025. arXiv: 2408.00166 [cs.IR]. URL: https://arxiv.org/abs/2408.00166.

[14] Julian McAuley et al. *Image-based Recommendations on Styles and Substitutes*. 2015. arXiv: 1506.04757 [cs.CV]. URL: https://arxiv.org/abs/1506.04757.

[15] Christian Meilicke et al. “Anytime bottom-up rule learning for knowledge graph completion”. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. IJCAI’19. Macao, China: AAAI Press, 2019, pp. 3137–3143. ISBN: 9780999241141.

[16] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL]. URL: https://arxiv.org/abs/1301.3781.

[17] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. “DeepWalk: online learning of social representations”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’14. ACM, Aug. 2014, pp. 701–710. DOI: 10.1145/2623330.2623732. URL: http://dx.doi.org/10.1145/2623330.2623732.

[18] Zhu Qiannan et al. “A Knowledge-Aware Attentional Reasoning Network for Recommendation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (Apr. 2020), pp. 6999–7006. DOI: 10.1609/aaai.v34i04.6184.

[19] Jiezhong Qiu et al. “DeepInf: Social Influence Prediction with Deep Learning”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*. KDD ’18. ACM, July 2018, pp. 2110–2119. DOI: 10.1145/3219819.3220077. URL: http://dx.doi.org/10.1145/3219819.3220077.

[20] Ruihong Qiu et al. “CausalRec: Causal Inference for Visual Debiasing in Visually-Aware Recommendation”. In: *Proceedings of the 29th ACM International Conference on Multimedia*. MM ’21. ACM, Oct. 2021, pp. 3844–3852. DOI: 10.1145/3474085.3475266. URL: http://dx.doi.org/10.1145/3474085.3475266.

[21] Steffen Rendle et al. *BPR: Bayesian Personalized Ranking from Implicit Feedback*. 2012. arXiv: 1205.2618 [cs.IR]. URL: https://arxiv.org/abs/1205.2618.

[22] Steffen Rendle et al. *BPR: Bayesian Personalized Ranking from Implicit Feedback*. 2012. arXiv: 1205.2618 [cs.IR]. URL: https://arxiv.org/abs/1205.2618.

[23] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *”Why Should I Trust You?”: Explaining the Predictions of Any Classifier*. 2016. arXiv: 1602.04938 [cs.LG]. URL: https://arxiv.org/abs/1602.04938.

[24] Ali Sadeghian et al. *DRUM: End-To-End Differentiable Rule Mining On Knowledge Graphs*. 2019. arXiv: 1911.00055 [cs.LG]. URL: https://arxiv.org/abs/1911.00055.

[25] Michael Schlichtkrull et al. *Modeling Relational Data with Graph Convolutional Networks*. 2017. arXiv: 1703.06103 [stat.ML]. URL: https://arxiv.org/abs/1703.06103.

[26] Yizhou Sun et al. “PathSim: meta path-based top-K similarity search in heterogeneous information networks”. In: *Proc. VLDB Endow.* 4.11 (Aug. 2011), pp. 992–1003. ISSN: 2150-8097. DOI: 10.14778/3402707.3402736. URL: https://doi.org/10.14778/3402707.3402736.

[27] Yizhou Sun et al. “PathSim: meta path-based top-K similarity search in heterogeneous information networks”. In: *Proc. VLDB Endow.* 4.11 (Aug. 2011), pp. 992–1003. ISSN: 2150-8097. DOI: 10.14778/3402707.3402736. URL: https://doi.org/10.14778/3402707.3402736.

[28] R.S. Sutton and A.G. Barto. “Reinforcement Learning: An Introduction”. In: *IEEE Transactions on Neural Networks* 9.5 (1998), pp. 1054–1054. DOI: 10.1109/TNN.1998.712192.

[29] Christos Troussas and Akrivi Krouska. “Path-Based Recommender System for Learning Activities Using Knowledge Graphs”. In: *Information* 14 (Dec. 2022), p. 9. DOI: 10.3390/info14010009.

[30] Shikhar Vashishth et al. *Composition-based Multi-Relational Graph Convolutional Networks*. 2020. arXiv: 1911.03082 [cs.LG]. URL: https://arxiv.org/abs/1911.03082.

[31] Hongwei Wang et al. “Knowledge Graph Convolutional Networks for Recommender Systems”. In: *The World Wide Web Conference*. WWW ’19. ACM, May 2019, pp. 3307–3313. DOI: 10.1145/3308558.3313417. URL: http://dx.doi.org/10.1145/3308558.3313417.

[32] Hongwei Wang et al. “Knowledge Graph Convolutional Networks for Recommender Systems”. In: *The World Wide Web Conference*. WWW ’19. San Francisco, CA, USA: Association for Computing Machinery, 2019, pp. 3307–3313. ISBN: 9781450366748. DOI: 10.1145/3308558.3313417. URL: https://doi.org/10.1145/3308558.3313417.

[33] Hongwei Wang et al. “RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems”. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM ’18. ACM, Oct. 2018, pp. 417–426. DOI: 10.1145/3269206.3271739. URL: http://dx.doi.org/10.1145/3269206.3271739.

[34] Xiang Wang et al. *Explainable Reasoning over Knowledge Graphs for Recommendation*. 2018. arXiv: 1811.04540 [cs.IR]. URL: https://arxiv.org/abs/1811.04540.

[35] Xiang Wang et al. *Explainable Reasoning over Knowledge Graphs for Recommendation*. 2018. arXiv: 1811.04540 [cs.IR]. URL: https://arxiv.org/abs/1811.04540.

[36] Yikun Xian et al. “CAFE: Coarse-to-Fine Neural Symbolic Reasoning for Explainable Recommendation”. In: *Proceedings of the 29th ACM International Conference on Information amp; Knowledge Management*. CIKM ’20. ACM, Oct. 2020, pp. 1645–1654. DOI: 10.1145/3340531.3412038. URL: http://dx.doi.org/10.1145/3340531.3412038.

[37] Yikun Xian et al. “Reinforcement Knowledge Graph Reasoning for Explainable Recommendation”. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR’19. Paris, France: Association for Computing Machinery, 2019, pp. 285–294. ISBN: 9781450361729. DOI: 10.1145/3331184.3331203. URL: https://doi.org/10.1145/3331184.3331203.

[38] Wenhan Xiong, Thien Hoang, and William Yang Wang. *DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning*. 2018. arXiv: 1707.06690 [cs.CL]. URL: https://arxiv.org/abs/1707.06690.

[39] Fan Yang, Zhilin Yang, and William W. Cohen. *Differentiable Learning of Logical Rules for Knowledge Base Reasoning*. 2017. arXiv: 1702.08367 [cs.AI]. URL: https://arxiv.org/abs/1702.08367.

[40] Jin-Cheng Zhang et al. “A review of recommender systems based on knowledge graph embedding”. In: *Expert Systems with Applications* 250 (2024), p. 123876. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2024.123876. URL: https://www.sciencedirect.com/science/article/pii/S0957417424007425.

[41] Yongfeng Zhang and Xu Chen. “Explainable Recommendation: A Survey and New Perspectives”. In: *Foundations and Trends® in Information Retrieval* 14.1 (2020), pp. 1–101. ISSN: 1554-0677. DOI: 10.1561/1500000066. URL: http://dx.doi.org/10.1561/1500000066.

[42] Yuyue Zhao et al. “Time-aware Path Reasoning on Knowledge Graph for Recommendation”. In: *ACM Transactions on Information Systems* 41.2 (Dec. 2022), pp. 1–26. ISSN: 1558-2868. DOI: 10.1145/3531267. URL: http://dx.doi.org/10.1145/3531267.

[43] Yaxin Zhu et al. *Faithfully Explainable Recommendation via Neural Logic Reasoning*. 2021. arXiv: 2104.07869 [cs.IR]. URL: https://arxiv.org/abs/2104.07869.