Path Language Modeling over Knowledge Graphs for Explainable Recommendation

Shijie Geng
Rutgers University
New Brunswick, NJ, US
sg1309@rutgers.edu

Zuohui Fu*
Meta Platforms, Inc.
Menlo Park, CA, US
zuohuif@fb.com

Juntao Tan
Rutgers University
New Brunswick, NJ, US
juntao.tan@rutgers.edu

Yingqiang Ge
Rutgers University
New Brunswick, NJ, US
yingqiang.ge@rutgers.edu

Gerard de Melo
HPI / University of Potsdam
Potsdam, Germany
gdm@demelo.org

Yongfeng Zhang
Rutgers University
New Brunswick, NJ, US
yongfeng.zhang@rutgers.edu

ABSTRACT

To facilitate human decisions with credible suggestions, personalized recommender systems should have the ability to generate corresponding explanations while making recommendations. Knowledge graphs (KG), which contain comprehensive information about users and products, are widely used to enable this. By reasoning over a KG in a node-by-node manner, existing explainable models provide a KG-grounded path for each user-recommended item. Such paths serve as an explanation and reflect the historical behavior pattern of the user. However, not all items can be reached following the connections within the constructed KG under finite hops. Hence, previous approaches are constrained by a *recall bias* in terms of existing connectivity of KG structures. To overcome this, we propose a novel Path Language Modeling Recommendation (PLM-Rec) framework, learning a language model over KG paths consisting of entities and edges. Through path sequence decoding, PLM-Rec unifies recommendation and explanation in a single step and fulfills them simultaneously. As a result, PLM-Rec not only captures the user behaviors but also eliminates the restriction to pre-existing KG connections, thereby alleviating the aforementioned recall bias. Moreover, the proposed technique makes it possible to conduct explainable recommendation even when the KG is sparse or possesses a large number of relations. Experiments and extensive ablation studies on three Amazon e-commerce datasets demonstrate the effectiveness and explainability of the PLM-Rec framework.

CCS CONCEPTS

• **Information systems** → *Recommender systems*; • **Computing methodologies** → *Knowledge representation and reasoning*.

*The paper was done when the author was at Rutgers University.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3511937

KEYWORDS

Path Language Model; Recommender Systems; Explainable Recommendation; Knowledge Graph; Recall Bias

ACM Reference Format:
Shijie Geng, Zuohui Fu, Juntao Tan, Yingqiang Ge, Gerard de Melo, and Yongfeng Zhang. 2022. Path Language Modeling over Knowledge Graphs for Explainable Recommendation. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3485447.3511937

1 INTRODUCTION

Explainable recommender systems have attracted increasing attention both in industry and in academia. Such systems not only bridge the gap between how the algorithm and customers perceive the relevance of items, but also open up the *black box* of the algorithmic decision process [56]. A prominent approach is that of explaining recommendations by means of knowledge graphs (KG) [1, 17, 23, 46, 48, 50, 51, 58, 59]. Such KGs [21] regard each user or item attribute as a vertex, and their relations are leveraged to build the graph edges (see Figure 1 for an example). Routes from a user to recommended items can serve as explanations, such that the relations along the path reflect a simulated decision-making process and thus provide explicit semantics for the explanations.

While existing KG-enhanced recommender systems have achieved many promising results, the KG paths used as explanations are usually produced in a post-hoc [1], pre-defined [48] or path-guided [50, 51, 59] manner, which entails a number of inherent weaknesses. In post-hoc approaches, the explanations are subsequently produced after a separate model predicts the recommended items. Since the generated explanations are not necessarily related with and may even be orthogonal to the recommendation decision process, they are not well-suited to promoting the system's transparency. Pre-defined methods (e.g., [48]) usually require enumerating all explanatory paths in an exhaustive manner before making final recommendation predictions, which is impractical in real-world recommender systems. Finally, existing approaches attempt to capture user behavior patterns and item-side knowledge originating from the KG to ensure the generalization ability in the inference stage. The latter can be achieved by either exploiting neural symbolic modules [51], adopting a reinforcement learning agent [50], or mining neural logic rules [59]. With the learned personalized rule and reward function, these methods conduct path-guided reasoning over KG.

However, all of the previous work remains limited to following the *truly connected* relations and edges in the KG to arrive at a reachable set of terminal items. In other words, the topology of the KG predetermines which items can be recommended, while a proportion of items may be entirely unreachable along any short multi-hop KG paths – we denote this phenomenon as ***recall bias***. Such recall bias may impede the applicability of these approaches in particular to KGs that are sparse or have a large number of relations.

We claim that an ideal KG explanation should accurately reveal the system's internal decision-making, i.e., how the system arrives at a recommendation, and the explanatory paths should be produced in an efficient manner to avoid combinatorial explosion. Moreover, a KG-based explainable approach should also gain the capability of inferring new paths beyond the limits of the static pre-constructed KG topology and better represent user behavior patterns, as it is not feasible in practical recommendation scenarios to offer explainable paths within the established KG based purely on historical user behavior and preference patterns. Fortunately, path language modeling is a promising avenue to achieve these goals in a unified manner. Recent work by Li et al. [28] learns an auto-regressive language model [33, 34] to predict the next edge or entity, given previous edges or entities in a path. In their work, they use the learned language model as a scoring function to rate the salience and coherence of a single path and select suitable path instances to construct a graph schema.

In our work, we train a path language model on possible KG paths between every user–item pair. However, unlike Li et al. [28], our trained path language model is used to predict novel paths and make recommendations rather than merely serving as a scoring function. With a user token as initial prompt, we generate potential paths with the learned path language model until a terminal <EOS> token is decoded or the hop limit is reached. By calculating the joint probability of each generated path, our Path Language Modeling Recommendation (PLM-Rec) selects items from high-scoring paths and makes recommendations. Thus, PLM-Rec can preserve the possibility of reaching even items that are unreachable through existing KG paths, thereby circumventing the recall bias in previous approaches. Additionally, PLM-Rec regards relations as a special type of edge tokens, which enables effortless scalability to a large number of relations. Last but not least, path-guided reasoning approaches [51, 59] typically require an additional *fine* stage to ground personalized rules to concrete KG paths. In contrast, PLM-Rec predicts next edge or entity tokens that naturally form a concrete path, obviating the need for any further processing steps. In summary, PLM-Rec merges reasoning and recommendation in a single stage and addresses all of the aforementioned shortcomings of previous approaches simultaneously.

In this work, we learn to capture user–item interactions through path language modeling over KG paths. We particularly seek to understand the following questions: 1) how to verify the concerns about recall bias in existing KG-based explainable recommendation approaches and quantify such recall bias; 2) how to perform path language modeling over a KG as well as unify the path reasoning and item recommendation processes into a shared path decoding stage; 3) how to infer paths beyond the predefined KG topology by means of suitable decoding strategy so as to enhance the generalization ability of the model. The key contributions of our paper can be outlined as follows:

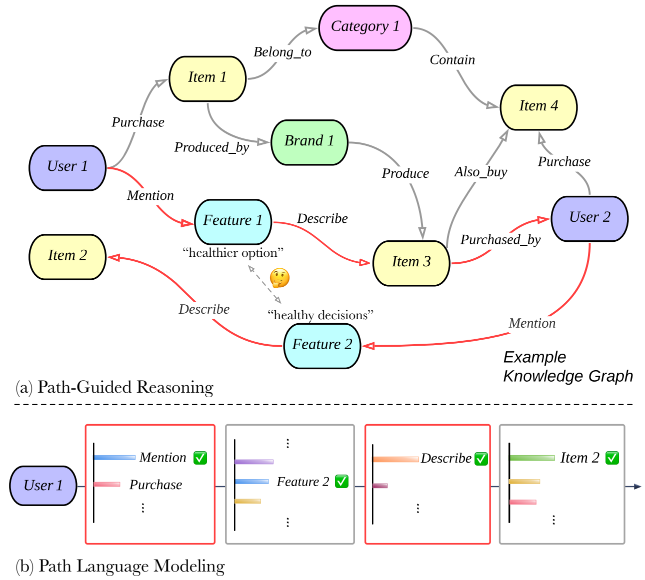


Figure 1: An example to illustrate the difference between a) traditional Path-Guided Reasoning approaches (relations in reasoning path shown as red lines) and b) the proposed Path Language Modeling Recommendation (relations in generated path sequence are shown in red boxes) when performing inference over knowledge graphs. PLM-Rec is able to go beyond the original KG topology by learning semantic relationships among features.

- We point out the shortcomings of previous KG-based explainable recommender systems where the newly discovered *recall bias* affects the recommendation performance but has been neglected in previous work.
- We conduct data-driven studies to prove the existence of recall bias in knowledge graphs and devise a new metric to quantify and evaluate such recall bias.
- We propose PLM-Rec to conduct reasoning over knowledge graphs for explainable recommendation, which learns to capture user behavior and item-side knowledge through path language modeling. As a result, PLM-Rec addresses the shortcomings in a unified framework, especially for the recall bias.
- Experiments on multiple real-world e-commerce recommendation datasets demonstrate that our approach outperforms several state-of-the-art baselines in terms of recommendation performance, while generating intuitive explanations.

2 RELATED WORK

Recommendation with Knowledge Graphs. Recommender Systems (RS) can be modeled as either a perceptual learning problem through Collaborative Filtering (CF) [16, 37] or a cognitive reasoning problem through Collaborative Reasoning (CR) [6, 7, 39]. Recently, it has become increasingly important to incorporate knowledge graph (KG) reasoning into recommender systems for both better performance and explainability. Previous efforts have attempted to make recommendations to users with the help of knowledge graph embeddings [2]. One research direction leverages knowledge graph embeddings as rich content information to enhance the recommendation performance. For example, Zhang et al. [55] adopted knowledge base embeddings to generate user and item representations, while Huang et al. [23] employed memory networks over knowledge graph entity embeddings for recommendation. Wang et al. [46] proposed a "ripple" network approach for embedding-guided multi-hop KG-based recommendation. Another research trend attempts to leverage the entity and path information in the knowledge graph to make explainable decisions. For example, Ai et al. [1] incorporate the learning of knowledge graph embeddings for explainable recommendation. However, their explanatory paths are essentially post-hoc explanations, as they are generated by soft matching after the corresponding items have been recommended. Wang et al. [48] proposed an RNN based model to reason over KGs for recommendation, which however requires enumerating all possible paths between each user–item pair for model training and prediction, which can be very time consuming for large-scale knowledge graphs. Xian et al. [50] formulate KG-based recommendation as a Markov Decision Process with path-based inference guided by learned policies. CAFE [51] further introduces a coarse-to-fine paradigm on the basis of neural symbolic reasoning for explicit user pattern modeling. LOGER [59] adopts neural logic reasoning to learn personalized rules with the help of the EM algorithm. However, these two methods both require an additional *fine* stage to ground user profiles to concrete paths. Our proposed approach is different from previous research in that it facilitates on-the-fly reasoning so that the recommendations are direct results of the explainable reasoning procedure. Meanwhile, there is no need to extract all paths between user–item pairs during inference, which makes the algorithm applicable to large-scale knowledge graphs.

Explainable Recommendation. Explainable recommendation has been an important task in both academia and industry [43, 44, 52, 56]. Early approaches predominantly attempt to make latent factor models explainable by aligning each latent dimension with an explicit features [9, 57]. With the rapid growth of deep learning technology, neural network components such as attention mechanisms were harnessed for improved explainability. User review text related to a user or item is concatenated to form a document, and by attentively seeking out valuable information within the document, highlighting the parts with the highest attention weights may serve as an explanation. For example, Seo et al. [38] attentively highlight particular words in user reviews as explanations, and Chen et al. [8, 10] proposed visually explainable recommendation by highlighting image regions. Based on natural language generation, recent research also generate natural language explanations for recommendation [5, 27]. In addition to text-based or image-based explainable recommendation, more recently, knowledge-aware explainable recommendation has attracted substantial research attention [1, 46, 50, 51, 59], as introduced in the previous subsection. HeteroEmbed [1] is a representative method, which conducts explainable recommendation by reasoning over knowledge graph embeddings, where the paths between a user and recommended items in the knowledge graph are considered as explanations.

Language Modeling over Paths. Language models [33] leverage various statistical or probabilistic techniques to establishes contextual rules and determine the probability of a given sequence of natural language words. Substantial progress has been achieved in recent years with neural auto-regressive language modeling [12, 32, 34], as most notably embodied in the success of the GPT series [4, 35], contextualized language modeling [11, 30, 54] as in the successful BERT [13] models, or a combination of both styles [14, 26, 40]. There are also attempts to incorporate KGs into pretrained language models [29, 42, 49] for improved entity-aware representations or better natural language generation [3, 18, 31, 53]. In the meantime, language models have also been applied to directly learn node representations over paths in a heterogeneous graph structure [15, 19, 20]. Recently, Li et al. [28] first proposed to employ a path language model for event graph schema induction, where a path in the KG is represented as a sequence of interleaved entity and edge tokens. The learned path language model is then utilized to score and select coherent and salient event schemas. Our proposed PLM-Rec approach shares a similar idea, but utilizes the learned path language model to not only capture user–item interactions but also generate the candidate path sequences with corresponding joint probabilities as ranking scores. Hence, the recommendation and path reasoning processes are unified in a single process. At the same time, PLM-Rec overcomes the constraints of the predefined KG topology by only considering the coherence and reasonableness of candidate paths.

Table 1: Recall bias statistics.

	Cellphones	Grocery	Automotive
3-hop	41%	37%	36%
4-hop	35%	29%	30%
5-hop	19%	12%	14%

3 PRELIMINARIES

In this section, we introduce basic concepts concerning the problem of KG-based Explainable Recommendation.

In general, there are two graphs available for KG-based recommendation tasks. One is the product graph containing attributes of all items $\mathcal{G}_{\mathcal{A}}$, which can be defined as $\mathcal{G}_{\mathcal{A}} = \{(e_h, r, e_t) \mid e_h, e_t \in \mathcal{E}_p, r \in \mathcal{R}_p\}$, where $\mathcal{E}_p$ is the entity set and $\mathcal{R}_p$ is the relation set. A triplet (e_h, r, e_t) indicates that the head entity e_h and the tail entity e_t are connected by the directed relation r. Another graph $\mathcal{G}_{\mathcal{U}\mathcal{V}}$ stores the user–item interaction data, which consists of two separate entity sets $\mathcal{U}$ and $\mathcal{V}$, and $\mathcal{E}_u = \mathcal{U} \cup \mathcal{V}$. The user entity $u \in \mathcal{U}$ and item entity $v \in \mathcal{V}$ are connected by a special interaction relation $r_{uv} \in \mathcal{R}_u$ when there is a *purchase* action in e-commerce or a *like* action in music recommendation for the user–item pair. By merging $\mathcal{G}_{\mathcal{A}}$ with $\mathcal{G}_{\mathcal{U}\mathcal{V}}$, the resulting user-centric *knowledge graph* $\mathcal{G}$ can be used for recommendation. In KG-based explainable recommendation, a *path* can be formally defined as a sequence S of entities and relations in KG: $S(e_0, e_l) = \{e_0, r_1, e_1, r_2, \cdots, r_l, e_l\}$, where $l+1$ entities $\{e_i\}_{i=0}^{l}$ are connected by l relations $\{r_j\}_{j=1}^{l}$. We call $S(e_0, e_l)$ an l-hop path that links head entity e_0 to tail entity e_l. In user-centric recommendation scenarios, a reasoning path $S(u, v)$ that originates from a user entity and ends at an item entity (*i.e.,* $e_0 \in \mathcal{U}$ and $e_l \in \mathcal{V}$) is eligible to serve as an explanation of recommendation. Specifically, the relation sub-sequence of $S(u, v)$ can be denoted as $\pi(u) = \{r_j\}_{j=1}^{l}$, which partly represents user u's preference and behavior pattern. Based on the aforementioned concepts and notations, we thus formalize the problem of *KG-based Explainable Recommendation (KG-ER)* as: Given a knowledge graph $\mathcal{G}$ and a test user u, the goal is to select a set of K recommendation items $\{v_k \mid v_k \in \mathcal{V}, (u, r_{uv_k}, v_k) \notin \mathcal{G}\}_{k=1}^{K}$ for user u along with K corresponding user-centric reasoning paths $\{S(u, v_k)\}_{k=1}^{K}$.

4 RECALL BIAS FOR KG PATHS

Once constructed, knowledge graphs are often considered as *gold standard* data sources. Prior approaches [1, 46, 50, 51, 59] follow the preexisting connections in KGs to perform path-guided reasoning and make recommendations. However, merely considering such truly connected paths may lead to important items never being reachable given the fixed lengths of reasoning paths, which limits the recall rate even before any training has taken place. The key question becomes how to formalize and quantify such statistical parity [41] in terms of the new reach ratio of ground-truth items that originated from the constructed KGs.

To quantify such recall bias, we straightforwardly leverage the ratio of unreachable items given the fixed lengths of reasoning paths to demonstrate the degree of recall bias arising in the datasets. Formally, we denote such items as $\hat{\mathcal{V}}$ and the **recall bias** can be calculated as $\frac{|\hat{\mathcal{V}}|}{|\mathcal{V}|}$, where the denominator is the number of ground truth items in the test dataset. To verify our claim, we first conduct a data-driven study to prove the existence of recall bias in KG-based explainable recommendation. Note that such post-hoc statistical experiments are designed only to verify our concerns for existing KG-based explainable recommendations and our model does not take such information as prior input. The results are given in Table 1, while dataset details can be found in Section 6.

We further design a corresponding metric called **new reach ratio (NR^2)** to measure to what extent the KG based explainable model mitigates the recall bias. Formally, it is defined as

$$\mathrm{NR}^2 = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\hat{V}_u|}{|V_u^{\mathrm{top}}|}, \tag{1}$$

where $|V_u^{\mathrm{top}}|$ denotes the number of ground truths items in top-K results of user u generated by recommender models. $|\hat{V}_u|$ stands for the number of *new* ground truths items within top-K predictions that are discovered by recommender models and $\hat{V}_u \in \hat{\mathcal{V}}$. Note that these items cannot be reached through path-finding in the original graph by any of the baseline methods.

5 PATH LANGUAGE MODELING FOR RECOMMENDATION

In this section, we first revisit previous paradigms for the KG-ER problem. Then we describe how to construct training path sequences and relevant sequence augmentation strategies. Subsequently, we introduce our proposed Path Language Modeling Recommendation (PLM-Rec) framework with details on the embedding layer, model, and training objective. Finally, we present how to decode candidate path sequences and make recommendation with the trained PLM-Rec model.

5.1 Overview of KG-ER problem

The KG-ER problem consists of two sub-tasks – making recommendations for each user and generating a path sequence as the explanation of each recommended item. We denote the two tasks as functions $f(\cdot)$ and $g(\cdot)$. Many approaches [1, 17, 46, 48, 50, 51, 59] have been proposed for solving the KG-ER problem. Typically, they follow three different paradigms during inference, namely post-hoc, pre-defined, or path-guided inference.

In the post-hoc paradigm, the recommendation is first made by calculating a similarity score between user and item embeddings trained with rich KG information: $\{v_k\}_{k=1}^{K} = f(u, \mathcal{V})$. Then, a separate path-finding process is conducted to retrieve explainable paths: $\{S(u, v_k)\}_{k=1}^{K} = g\left(u, \{v_k\}_{k=1}^{K}, \mathcal{G}\right)$. In the pre-defined paradigm, all paths that connect interacting user–item pairs are first extracted from $\mathcal{G}$ through a search algorithm such as breadth-first search. For a random user u, we assume there are M retrieved paths: $\{S(u, v_i)\}_{i=1}^{M} = g(u, \mathcal{G})$. A pretrained path scoring model is finally invoked to select prominent paths and make recommendations: $\{v_k, S(u, v_k)\}_{k=1}^{K} = f\left(\{S(u, v_i)\}_{i=1}^{M}\right)$. Starting from a user u, path-guided approaches directly move along KG connections until reaching a final item v. The whole process can be represented as $\{v_k, S(u, v_k)\}_{k=1}^{K} = g(u, \mathcal{G})$. Nevertheless, all above paradigms are restricted to pre-existing paths in the constructed KG. In fact, $\mathcal{G}$ is built with historical data and is usually incomplete. Under a limited path length, some potential items can never be reached from a given user. Thus, it is necessary to relax the requirement of path tracking and allow the exploration of new connections across nodes in the KG. This approach is at least as powerful as increasing the maximum path length in the aforementioned paradigms.

In contrast to prior approaches, our PLM-Rec framework can alleviate the recall bias and achieves explainable recommendation in a single unified step. After training a path language model $\phi(\cdot)$, we directly apply it to generate candidate path sequences for each user: $\{v_k, S(u, v_k)\}_{k=1}^{K} = \phi(u)$, where the relations in $S(u, v_k)$ may either be original links in the KG or novel ones.

5.2 Path Language Modeling over KGs

Training Path Sequence Construction. Knowledge graphs in e-commerce are massive and informative, comprising a large number of entities and relations. For the purpose of training path language models, we require large amounts of path sequences that both represent user behavior pattern and item-side knowledge. To this end, we first employ an off-the-shelf random walk algorithm used in previous work [25] to extract training paths from $\mathcal{G}$ with maximum number of hops N. The training paths satisfy two requirements: 1) starting from a user entity u and ending at an item entity v. 2) u and v are connected by an interaction relation r_{uv} in the user–item interaction graph $\mathcal{G}_{\mathcal{U}\mathcal{V}}$. Without loss of generality, we denote an l-hop extracted path as $S_{\mathrm{train}} = [u, r_1, e_1, r_2, \cdots, e_{l-1}, r_l, v]$. Although the original training paths include enough historical records regarding users and items, they are insufficiently diverse to explore new possibilities within the KG. To further improve the robustness of the trained PLM-Rec model, it is necessary to augment the input sequences. After examining the constructed KG, we made the following observations: 1) User reviews typically contain descriptions of the features of purchased items, and these features account for

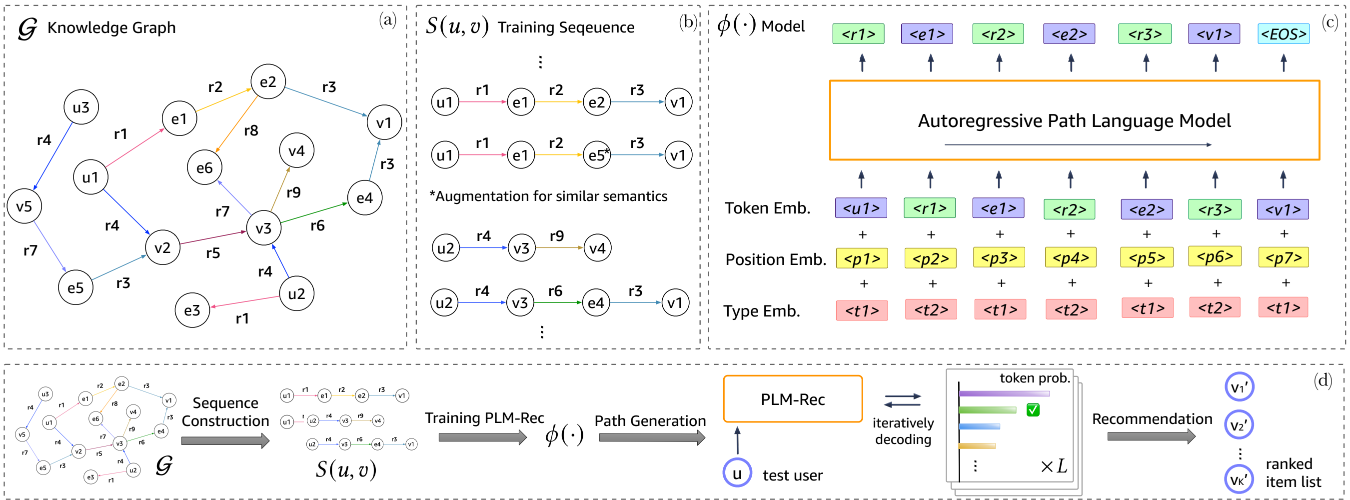


Figure 2: Overview of PLM-Rec framework. (a) shows an example knowledge graph $\mathcal{G}$, from which we extract training path sequences under different hop constraints. By leveraging augmentations for features with similar semantics, we achieve a series of training data $S(u, v)$ in (b). We adopt a Transformer-based decoder to train an autoregressive path language model $\phi(\cdot)$ in (c). The entire pipeline including path generation and recommendation steps for test users is presented in (d).

a large proportion of all entities; 2) due to the scale of $\mathcal{G}$, there are many similar but distinct style or feature entities. Hence, we perform sequence augmentation by randomly substituting a style or feature entity with a similar entity. We use Sentence-BERT [36] as a metric to measure the semantic similarity and set a similarity threshold θ to decide augmentation candidates for all style and feature entities. The union of original and augmented path sequences serves as the final overall training data.

Token Types and Embeddings. In PLM-Rec, we define two basic types of input tokens – entity tokens and relation tokens. We denote their embeddings as $\mathcal{E}_e \in \mathbb{R}^{|\mathcal{E}|\times d}$ and $\mathcal{E}_r \in \mathbb{R}^{|\mathcal{R}|\times d}$, where d is the dimensionality of embeddings. The embedding of $\mathcal{S}_{\text{train}}$ can be represented as $\mathcal{S}_{\text{train}} = [\mathbf{u}, \mathbf{r}_1, \mathbf{e}_1, \mathbf{r}_2, \cdots, \mathbf{e}_{l-1}, \mathbf{r}_l, \mathbf{v}]$. The generation of path sequences follows the autoregressive style [33], i.e., decoding one token at a time from left to right given the sequence of previously observed tokens. The generation is triggered by the initial user entity token u, upon which we alternatively decode relation tokens and entity tokens until reaching the special <EOS> token, which indicates the termination of the sequence. The maximum hop number N implies that the maximum length L of desired path sequences is $L = 2N + 1$. If the number of generated text tokens before <EOS> is less than L, we pad the sequence with another special token <PAD>. Before feeding the token sequence into PLM-Rec, we add positional embeddings $\mathcal{P} \in \mathbb{R}^{L\times d}$ to the raw embeddings, capturing the position within the sequence. Furthermore, we also add a type embedding $\mathcal{T} = [t_1, t_2, t_1, \cdots, t_2, t_1, t_0] \in \mathbb{R}^{L\times d}$ to help distinguish entities and relations. Specifically, t_1, t_2, t_0 stands for entities, relations, and special tokens respectively. The final input sequence representation $\mathbf{S}_0$ can be written as

$$\mathbf{S}_0 = \mathcal{S}_{\text{train}} + \mathcal{P} + \mathcal{T} \tag{2}$$

Autoregressive Path Language Model. To capture the probability distribution of entity and relation tokens, we adopt Transformer [45] decoder layers to learn the autoregressive path language model. Suppose the Transformer decoder layer has h heads in multi-head self-attention. For input sequence $\mathbf{S}_i$ at layer $i \in [0, 1, \ldots, l]$, the encoded sequence $\mathbf{S}_{i+1}$ can be computed as

$$\mathbf{S}_{i+1} = \text{FFN}_i \left(\text{Attention}\left(\mathbf{S}_i\mathbf{W}_{\text{Q}}, \mathbf{S}_i\mathbf{W}_{\text{K}}, \mathbf{S}_i\mathbf{W}_{\text{V}}\right)\right)$$

Here, $\mathbf{W}_{\text{Q}}, \mathbf{W}_{\text{K}}, \mathbf{W}_{\text{V}} \in \mathbb{R}^{d\times d_h}$ are weight matrices for projecting query, key, and value respectively [45], $d_h = d/h$ is the dimensionality for each head. FFN_i is a feed-forward module consisting of two fully-connected layers with ReLU activation. The Attention function is defined as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}}\right)\mathbf{V}$$

with a scaling factor $\sqrt{d_h}$ that maintains the order of magnitude in features. We adopt a traditional causal masking strategy during training to avoid any leakage of future information [45].

Finally, we predict next tokens with two separate fully-connected layers. For entity and relation tokens, we have

$$P\left(e_j \mid u, \ldots, r_j\right) = \text{Softmax}\left(\mathbf{S}_l W_e + b_e\right) \tag{3}$$

$$P\left(r_{j+1} \mid u, r_1, \ldots, e_j\right) = \text{Softmax}\left(\mathbf{S}_l W_r + b_r\right) \tag{4}$$

to estimate the token probability distribution, respectively. W_e, W_r, b_e, b_r are weight matrices and bias vectors of the two fully-connected layers. Then the language modeling loss for path sequence generation can be expressed as

$$\mathcal{L}_{PLM} = \sum_{\mathcal{S}_{\text{train}}} \Bigg[\sum_{e_j \in \mathcal{S}_{\text{train}}} \log P\left(e_j \mid u, \ldots, r_j\right) + \sum_{r_{j+1} \in \mathcal{S}_{\text{train}}} \log P\left(r_{j+1} \mid u, r_1, \ldots, e_j\right) \Bigg] \tag{5}$$

5.3 Path Generation and Recommendation

Decoding Path Sequences. With the pretrained PLM-Rec model $\phi(\cdot)$ and an initial user token u, we can conduct path generation under certain decoding strategy. The goal of explainable recommendation is to select the list $\{v_k, S(u, v_k)\}_{k=1}^{K}$, so greedy search is unsuitable, since only one sequence would be decoded. In this paper, we adopt Nucleus Sampling [22] as the strategy for decoding path sequences. We define the nucleus probability to be p. Thus, we can select a top-p vocabulary $V^{(p)} \subset V$ at each decoding step as

$$\sum_{S_i \in V^{(p)}} P\left(S_i \mid S_{1:i-1}\right) \geq p \tag{6}$$

Based on $V^{(p)}$, we re-scale the original distribution to a new one, and then sample the next word from it:

$$P'\left(S_i \mid S_{1:i-1}\right) = \begin{cases} P\left(S_i \mid S_{1:i-1}\right)/p' & \text{if } S_i \in V^{(p)} \\ 0 & \text{otherwise.} \end{cases}, \tag{7}$$

where $p' = \sum_{S_i \in V^{(p)}} P\left(S_i \mid S_{1:i-1}\right)$. To make recommendations, we usually sample $T > K$ sequences from the learned path language model, and finally select K sequences along with corresponding items by ranking the joint probability $P(u, r_1, e_1, \cdots, r_l, v)$.

6 EXPERIMENTS

In this section, we evaluate the performance of the proposed PLM-Rec approach on real-world datasets compared with seven representative and state-of-the-art recommendation methods. We aim to answer the following research questions:

- **RQ1:** How does PLM-Rec perform compared with state-of-the-art knowledge-based recommendation methods?
- **RQ2:** How do factors such as the path length, sequence augmentation, decoding strategy affect the performance of PLM-Rec?
- **RQ3:** Can PLM-Rec provide reasonable explanations about user preferences towards certain recommended items?

Table 2: Basic statistics of the experimental datasets.

Dataset	**Cellphones**	**Grocery**	**Automotive**
#Users	61,254	57,822	95,445
#Items	47,604	40,694	78,557
#Interactions	607,673	709,280	1,122,776
Sparsity (%)	0.0208	0.0301	0.0150
#Entities	169,331	173,369	270,543
#Relations	45	45	73
#Triples	3,117,051	3,742,954	4,580,318

6.1 Experimental Setup

Datasets. We adopt the consumer transaction dataset crawled from Amazon.com[1] in our experiments. The dataset includes user reviews & transactions (user_id, item_id, rating, user review, etc.) and item metadata (item_id, price, related_items, category, brand, etc.) on 24 product categories dated from May 1996 to July 2014. We take three categories (*Cellphone*, *Grocery*, *Automotive*) of different entity and relation sizes to validate the performance of our model. We utilize the Amazon e-commerce dataset in that it is a large-scale dataset for e-commerce recommendation, which contains rich user behavior patterns (such as mentioned feature words, preferred styles, etc.) and item-side knowledge (such as brands, categories, related products, etc.), and thus enables the creation of a large-scale product knowledge graph to generate recommendations together with corresponding explanations. In the constructed knowledge graphs, each user entity is connected to the item entities that they interacted with before through a *purchase* relation, and each item is connected to its brand/category/feature as well as related items through other attribute relations. A path is considered valid as long as it connects a user with the recommended item. As indicated in previous work [17, 48], greater path lengths introduce more noisy entities, while shorter paths tend to be more reliable for users as explanations of recommended items. Hence, we only take into consideration paths with up to 5 hops of relationships throughout our experiments. More detailed statistics of the three datasets are reported in Table 2. On each dataset, we split the records into training, valid, and testing splits chronologically with a ratio of 3:1:1, which follows the setups of [59]. Note that since the entities and relations of the knowledge graphs are from different domains, the evaluation results are not comparable across different KGs.

[1] https://nijianmo.github.io/amazon/

Implementation Details. In our experiments, the entity embedding size d is set to 100, and the entity embeddings are initialized with pretrained KG embeddings [2]. Besides user and item entities, other entities can be divided into seven types, i.e., aspect_value, feature, price, brand, category, related_product, style. For training sequence construction, the path hop N is set to 3 and the semantic similarity threshold θ is set to 0.8. The path language model is trained with Adam optimization [24] on Nvidia GeForce RTX 3090 GPU. In the training process, we adopt a learning rate of 2×10^{-4}, a batch size of 128, and a number of training epochs of 20. During path sequence decoding, we employ nucleus sampling with nucleus $p = 0.4$ as the basic strategy. The influence of these hyperparameters will be studied in Sections 6.3 through 6.5.

Baselines. We compare our model with the following baselines, including KG-based embedding approaches (CKE, RippleNet, KGAT) and Path-based reasoning approaches (HeteroEmbed, PGPR, CAFE, LOGER). Specifically, **CKE** [55], also known as the Collaborative Knowledge-base Embedding model, is a neural model that incorporates text, images, and a knowledge base for recommendation. **RippleNet** [46] follows the idea of user preference propagation to extend users' historical interests along KG links to facilitate recommendation. **KGAT** [47] leverages a graph-based attention network to capture high-order KG relations to improve recommendation performance. As for path-reasoning approaches, **HeteroEmbed** [1] first conducts recommendation based on pretrained TransE [2] entity embeddings and then performs post-hoc path searching over the KG to extract explanations for user-recommended items. **PGPR** [50] introduces reinforcement learning to learn a suitable multi-hop scoring function and path-reasoning policies in support of explainable recommendation. **CAFE** [51] adopts neural symbolic reasoning to achieve explainable recommendation. It first generates user behavior profiles in a *coarse* stage and then carries out path reasoning with the extracted profiles for KG grounding in a *fine* stage. **LOGER** [59] draws on logical rules to guide the path

Table 3: Recommendation performance of our method compared to other baselines on three Amazon datasets: Cellphones, Grocery, and Automotive. We follow four representative recommendation metrics (Precision, Recall, Hit Ratio, and NDCG) to evaluate the performance of different approaches and set the length of the recommendation list K to 10. The best results in each column are highlighted in bold, while underlined numbers denote second-best results.

	Cellphones				**Grocery**				**Automotive**			
	Precision	Recall	NDCG	HR	Precision	Recall	NDCG	HR	Precision	Recall	NDCG	HR
CKE	0.0360	0.1760	0.1847	0.3067	0.0612	0.2528	0.3070	0.4511	0.0458	0.1871	0.2257	0.3621
RippleNet	0.0419	0.2141	0.2177	0.3715	0.0591	0.2682	0.2858	0.4800	0.0477	0.1950	0.2353	0.3916
KGAT	0.0476	0.2274	0.2365	0.3835	0.0702	0.2916	0.3381	0.5020	0.0601	0.2500	0.2859	0.4514
PGPR	0.0462	0.2148	0.2366	0.3801	0.0649	0.2710	0.3174	0.4926	0.0589	0.2315	0.2804	0.4409
HeteroEmbed	0.0527	0.2543	0.2626	0.4226	0.0785	0.3316	0.3701	0.5572	0.0695	0.2923	0.3314	0.5082
CAFE	0.0608	0.2806	0.2995	0.4371	0.0823	0.3532	0.4016	0.5838	0.0717	0.2978	0.3475	0.5193
LOGER	<u>0.0622</u>	<u>0.2977</u>	<u>0.3227</u>	<u>0.4808</u>	<u>0.0906</u>	<u>0.3754</u>	<u>0.4370</u>	<u>0.6121</u>	<u>0.0743</u>	<u>0.3091</u>	<u>0.3653</u>	<u>0.5346</u>
PLM-Rec	**0.0642**	**0.3035**	**0.3423**	**0.4952**	**0.0933**	**0.3829**	**0.4479**	**0.6251**	**0.0776**	**0.3251**	**0.3818**	**0.5527**

reasoning, where personalized rules are learned through the EM algorithm in an iterative manner.

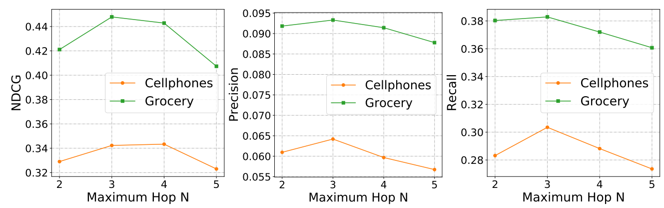
Table 4: Comparison of ability of different approaches to mitigate recall bias, in terms of NR^2.

Method	**Cellphones**	**Grocery**	**Automotive**
Path-guided methods	0	0	0
PLM-Rec ($N = 3$)	0.1656	0.1526	0.1904
PLM-Rec ($N = 4$)	0.1370	0.1238	0.1357
PLM-Rec ($N = 5$)	0.0668	0.0592	0.0929

Evaluation Metrics. We adopt the same four representative recommendation metrics as in previous work [59] to evaluate the models: Hit Ratio (**HR**), **Recall**, Precision (**Prec.**), and Normalized Discounted Cumulative Gain (**NDCG**). Meanwhile, we adopt the proposed New Reach Ratio (**NR^2**) to measure a model's ability to mitigate recall bias. Due to computational overhead, we randomly sample a user subset when calculating the value of NR^2.

6.2 Performance Comparison (RQ1)

We compare the performance of our model with baseline methods in terms of their top-10 recommendations. The main results under four recommendation metrics are reported in Table 3. Generally, our method achieves the best recommendation performance against both KG-based embedding and Path-based reasoning approaches across all settings. Taking the *Cellphones* dataset as an example, PLM-Rec obtains an NDCG@10 score of 0.3423, which improves over HeteroEmbed by 7.97%, CAFE by 4.28%, and the best baseline LOGER by 1.96%. Moreover, we observe that PLM-Rec shows better recall than baselines, while maintaining a higher precision. Similar trends can be observed on other benchmarks as well. Interestingly, we observe that our method is much better than the baselines on the *Automotive* dataset. Table 2 shows that the density of *Automotive* is the lowest compared with the other two datasets, which means that the *Automotive* dataset includes much fewer user–item interactions. The observation implies the advantage of our PLM-Rec approach on sparse datasets, owing to its ability to discover more latent connections across entities by exploring the semantic space of the KG. At the same time, *Automotive* also includes more relation types, which further shows PLM-Rec's capability of dealing with more relations. From Table 4, we find that PLM-Rec significantly breaks the constraint of KG topology under a fixed path length. In contrast to path-guided approaches, PLM-Rec effectively mitigates the recall bias and arrives at a greater number of ground truth items.



(a) NDCG (b) Precision (c) Recall

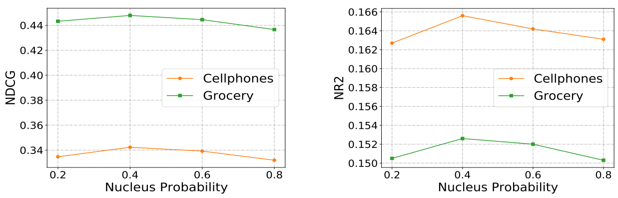
Figure 3: Results of varying maximum path hop length N on Cellphones (orange) and Grocery (green) datasets.

6.3 Ablation on Path Length (RQ2)

In this section, we evaluate how the maximum number of path hops N affects our model in the following two respects: 1) the performance in terms of the basic metrics (NDCG, Recall, Precision); 2) how many new items can be reached compared to previous path-guided approaches. According to Figure 3, almost all traditional recommendation metrics reach peak values when N is set to 3. In most cases, the scores for the three metrics rise as N increases from 2 to 3 and decrease when N continues increasing. These results demonstrate that 3 is the optimal choice of N on the Amazon datasets, which is in line with previous work, since longer-hop paths can introduce more noisy entities and weaken the performance of the path language model. From Table 4, we can observe that PLM-Rec generally mitigates more recall bias and achieves a higher NR^2 with shorter hop settings or in a sparser KG. This matches the statistics in Table 1 – a larger number of hops yields a greater number of reachable items.

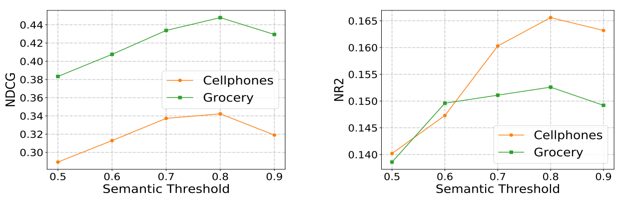
6.4 Ablation on Sequence Decoding (RQ2)

In this section, we study the influence of nucleus probability p during sequence decoding. We vary the nucleus probability p among $\{0.2, 0.4, 0.6, 0.8\}$. As shown in Figure 4, an optimal choice of probability p is usually around 0.4. If p is too large, more noisy tokens are sampled. If p is too small, the diversity of generation is affected, resulting in sub-optimal performance.



(a) NDCG (b) NR^2

Figure 4: Results of varying nucleus probability p on Cellphones (orange) and Grocery (green) datasets.



(a) NDCG (b) NR^2

Figure 5: Results of varying semantic threshold θ on Cellphones (orange) and Grocery (green) datasets.

6.5 Ablation on Sequence Augmentation (RQ2)

In this section, we discuss the influence of similarity threshold θ. Particularly, we try different θ amongst $\{0.5, 0.6, 0.7, 0.8, 0.9\}$ and plot how the NDCG and NR^2 scores develop according to different θ values in Figure 5. We find that 0.8 is the optimal choice of θ with regard to both NDCG and NR^2. Lower θ values lead to conflicting training sequences being generated, which may mislead PLM-Rec, while higher θ values imply less augmentation and decrease the generalization ability of PLM-Rec.

6.6 Case Study (RQ3)

Finally, we conduct case studies of the recommendations given by our PLM-Rec framework. Figure 6 provides two real-world examples from the *Grocery* dataset to illustrate how our model mitigates the recall bias through path sequence decoding.

The first one shows how PLM-Rec discovers the ground truth item *walnuts*, which is unreachable with path-finding under a 3-hop setting. The user has previously purchased a lemonade with feature *California-sourced citrus*. In the original KG topology, it is impossible to reach *walnuts* in 3 steps. However, *California-sourced citrus* suggests a potential preference of the user for California-produced products. Our PLM-Rec captures this latent behavior pattern and generates other California-related features and thus recommends *walnuts* to the user. This demonstrates PLM-Rec's ability to learn semantics and infer shortcuts to further recommendation items.

The second case involves recommending *matcha*-related utensils to a user who loves strong matcha flavor. Since matcha utensils such as *matcha tea set - b* belong to the *Home and Kitchen* category on Amazon, they serve as the tail entity of *also-buy* in the *Grocery* subset and there is no direct connection between *Home and Kitchen* items provided in *Grocery*. Therefore, in this case, a longer 6-hop path would be needed for the user entity to reach the item *matcha tea set - b* if following the existing KG links. In contrast, PLM-Rec

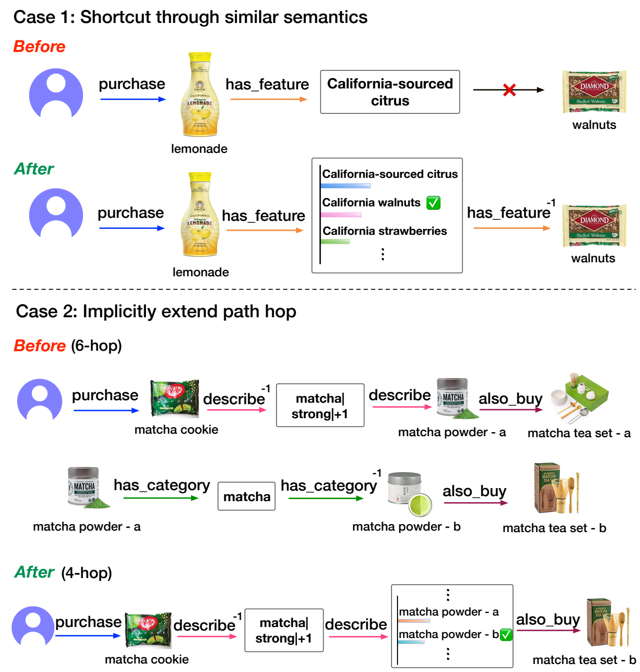


Figure 6: Case study of real-world recommendation examples, demonstrating two advantages of path language modeling in solving recall bias.

absorbs knowledge about products of the same category during training and thus only requires 4 hops to reach the recommended item. In other words, PLM-Rec can implicitly extend path hops by acquiring semantic generalization capabilities.

7 CONCLUSIONS

In this paper, we shed light on the problem of recall bias in prior approaches and explore the new direction of leveraging path language modeling to capture the knowledge and long-range dependencies along KG paths. PLM-Rec overcomes the constraints of sticking to pre-existing connections in the KG topology and thus eliminates the recall bias that past paradigms entail. Through a suitable decoding strategy, the learned path language model performs path-guided reasoning over knowledge graphs to simultaneously generate recommendations and corresponding explanations. It also unifies recommendation and path-based reasoning in a single step, thus avoiding the additional path grounding step of prior work. Experimental results including extensive ablation studies on three real-world datasets prove the effectiveness and generalization ability of our PLM-Rec model in terms of the recommendation performance as well as providing reasonable path-based explanations.

ACKNOWLEDGMENTS

We appreciate the valuable feedback and suggestions of the reviewers. This work was supported in part by NSF IIS 1910154, 2007907, and 2046457. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* (2018).

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

[3] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense Transformers for Automatic Knowledge Graph Construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4762–4779.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.

[5] Hanxiong Chen, Xu Chen, Shaoyun Shi, and Yongfeng Zhang. 2019. Generate natural language explanations for recommendation. *SIGIR 2019 Workshop on ExplainAble Recommendation and Search (EARS)* (2019).

[6] Hanxiong Chen, Yunqi Li, Shaoyun Shi, Shuchang Liu, He Zhu, and Yongfeng Zhang. 2022. Graph Collaborative Reasoning. *WSDM* (2022).

[7] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. In *Proceedings of the Web Conference 2021*. 1516–1527.

[8] Xu Chen, Hanxiong Chen, Hongteng Xu, Yongfeng Zhang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2019. Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 765–774.

[9] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to rank features for recommendation over multiple categories. In *SIGIR*.

[10] Xu Chen, Yongfeng Zhang, Hongteng Xu, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Visually explainable recommendation. *arXiv* (2018).

[11] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *ACL*.

[12] Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. *Advances in neural information processing systems* (2015).

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[14] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. In *NeurIPS*.

[15] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 135–144.

[16] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. 2011. *Collaborative filtering recommender systems*. Now Publishers Inc.

[17] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, et al. 2020. Fairness-aware explainable recommendation over knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 69–78.

[18] Shijie Geng, Peng Gao, Moitreya Chatterjee, Chiori Hori, Jonathan Le Roux, Yongfeng Zhang, Hongsheng Li, and Anoop Cherian. 2021. Dynamic Graph Representation Learning for Video Dialog via Multi-Modal Shuffled Transformers. In *AAAI*.

[19] Josu Goikoetxea, Aitor Soroa, and Eneko Agirre. 2015. Random walks and neural network language models on knowledge bases. In *Proceedings of the 2015 conference of the North American Chapter of the Association for Computational Linguistics: Human language technologies*. 1434–1439.

[20] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[21] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. *ACM Computing Surveys (CSUR)* (2021).

[22] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*.

[23] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*.

[24] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR* (2015).

[25] Ni Lao, Tom Mitchell, and William Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 conference on empirical methods in natural language processing*. 529–539.

[26] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7871–7880.

[27] Lei Li, Yongfeng Zhang, and Li Chen. 2021. Personalized Transformer for Explainable Recommendation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 4947–4957.

[28] Manling Li, Qi Zeng, Ying Lin, Kyunghyun Cho, Heng Ji, Jonathan May, Nathanael Chambers, and Clare Voss. 2020. Connecting the dots: Event graph schema induction with path language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 684–695.

[29] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[31] Ye Liu, Yao Wan, Lifang He, Hao Peng, and S Yu Philip. 2021. KG-BART: Knowledge Graph-Augmented BART for Generative Commonsense Reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[32] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

[33] Jay M Ponte and W Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 275–281.

[34] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018).

[35] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* (2019).

[36] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[37] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 291–324.

[38] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *RecSys*.

[39] Shaoyun Shi, Hanxiong Chen, Weizhi Ma, Jiaxin Mao, Min Zhang, and Yongfeng Zhang. 2020. Neural logic reasoning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1365–1374.

[40] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450* (2019).

[41] Harald Steck. 2011. Item popularity and recommendation accuracy. In *Proceedings of the fifth ACM conference on Recommender systems*. 125–132.

[42] Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. 2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137* (2021).

[43] Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. 2022. Learning and Evaluating Graph Neural Network Explanations based on Counterfactual and Factual Reasoning. *WWW* (2022).

[44] Juntao Tan, Shuyuan Xu, Yingqiang Ge, Yunqi Li, Xu Chen, and Yongfeng Zhang. 2021. Counterfactual Explainable Recommendation. *CIKM* (2021).

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

[46] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM*.

[47] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.

[48] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. *AAAI* (2019).

[49] Peter West, Chandra Bhagavatula, Jack Hessel, Jena D Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2021. Symbolic knowledge distillation: from general language models to commonsense models. *arXiv preprint arXiv:2110.07178* (2021).

[50] Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 285–294.

[51] Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard De Melo, Shan Muthukrishnan, et al. 2020. CAFE: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1645–1654.

[52] Shuyuan Xu, Yunqi Li, Shuchang Liu, Zuohui Fu, Yingqiang Ge, Xu Chen, and Yongfeng Zhang. 2021. Learning causal explanations for recommendation.

[53] Pengcheng Yang, Lei Li, Fuli Luo, Tianyu Liu, and Xu Sun. 2019. Enhancing topic-to-essay generation with external commonsense knowledge. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2002–2012.

[54] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).

[55] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*.

[56] Yongfeng Zhang, Xu Chen, et al. 2020. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.

[57] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*.

[58] Kangzhi Zhao, Xiting Wang, Yuren Zhang, Li Zhao, Zheng Liu, Chunxiao Xing, and Xing Xie. 2020. Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 239–248.

[59] Yaxin Zhu, Yikun Xian, Zuohui Fu, Gerard de Melo, and Yongfeng Zhang. 2021. Faithfully Explainable Recommendation via Neural Logic Reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3083–3090.