KGGLM: A Generative Language Model for Generalizable Knowledge Graph Representation Learning in Recommendation

Giacomo Balloccu
Department of Mathematics and Computer Science, University of Cagliari
Italy
gballoccu@acm.org

Ludovico Boratto
Department of Mathematics and Computer Science, University of Cagliari
Italy
ludovico.boratto@acm.org

Gianni Fenu
Department of Mathematics and Computer Science, University of Cagliari
Italy
fenu@unica.it

Mirko Marras
Department of Mathematics and Computer Science, University of Cagliari
Italy
mirko.marras@acm.org

Alessandro Soccol
Department of Mathematics and Computer Science, University of Cagliari
Italy
a.soccol@studenti.unica.it

Abstract

Current recommendation methods based on knowledge graphs rely on entity and relation representations for several steps along the pipeline, with knowledge completion and path reasoning being the most influential. Despite their similarities, the most effective representation methods for these steps differ, leading to inefficiencies, limited representativeness, and reduced interpretability. In this paper, we introduce KGGLM, a decoder-only Transformer model designed for generalizable knowledge representation learning to support recommendation. The model is trained on generic paths sampled from the knowledge graph to capture foundational patterns, and then fine-tuned on paths specific of the downstream step (knowledge completion and path reasoning in our case). Experiments on ML1M and LFM1M show that KGGLM beats twenty-two baselines in effectiveness under both knowledge completion and recommendation. Source code and pre-processed data sets are available at https://github.com/mirkomarras/kgglm.

CCS Concepts

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

Keywords

Knowledge Graph, Knowledge Graph Embeddings, Knowledge Representation Learning, Knowledge Completion, Recommendation, Language Model, Generative Artificial Intelligence.

ACM Reference Format:

Giacomo Balloccu, Ludovico Boratto, Gianni Fenu, Mirko Marras, and Alessandro Soccol. 2024. KGGLM: A Generative Language Model for Generalizable Knowledge Graph Representation Learning in Recommendation. In *18th ACM Conference on Recommender Systems (RecSys '24), October 14–18, 2024, Bari, Italy.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3640457.3691703

RecSys '24, October 14–18, 2024, Bari, Italy

ACM ISBN 979-8-4007-0505-2/24/10
https://doi.org/10.1145/3640457.3691703

1 Introduction

Incorporating knowledge graphs into recommendation has enhanced both their utility and explainability [4, 5, 21, 38]. The completeness of knowledge graphs is evidently crucial for realizing these improvements [14]. Many knowledge graphs are however incomplete, so enriching their initial version is a core step in the recommendation pipeline [8]. General-purpose methods for knowledge representation learning, which embed graph components into continuous vector spaces, are commonly used to address this effectively [2, 6, 9–11, 15–20, 28, 31, 34, 37]. Of these, tensor decomposition methods like [31, 37] decompose the graph into entity and relation embeddings, using either bi-linear or non-bi-linear scoring functions. Geometric methods like [6, 28] treat relations as transformations in a latent space, scoring links based on the distance between transformed embeddings. Deep learning methods like [10] use neural networks to learn patterns and produce final scores.

The same knowledge representation learning methods can be extended to predict links of potential interest between users and products during the path reasoning step [25]. However, these general-purpose methods often lack the ability to discover multi-hop relational patterns essential for predicting user preferences. To address this, multi-hop methods have been specialized for the path reasoning step by leveraging the pre-computed general-purpose knowledge graph embeddings [1, 12, 29, 33, 35, 36, 39]. This is achieved through various approaches, such as combining graph embeddings with collaborative filtering [39], using attention mechanisms to propagate embeddings [33], performing coarse-to-fine reasoning over embeddings [36], guiding agents within the embedding space [35], and integrating path-guided reasoning with explanatory graph features [12]. However, such specialization renders these methods ineffective for knowledge completion.

As a result, the most effective methods for knowledge representation learning at each step of the recommendation pipeline differ

fundamentally. This leads to inefficiencies due to the need for separate training and storage for each step, limited representativeness because the models used for knowledge completion do not align with those used for path reasoning, and reduced interpretability because the representations lie in substantially different areas of the continuous space. Our work advances beyond these limitations by leveraging the intuition that language models (e.g., GPT-2 [22]), effective at learning to predict text sequences, can be applied to learn cross-step generalizable representations over knowledge graphs.

In this paper, we introduce a method for learning generalizable knowledge graph representations, called KGGLM. Our approach involves using a random walk algorithm to sample paths from the knowledge graph, after being extended with user-product relations. We tokenize these paths into sequences and embed these sequences. We train a model based on Transformer decoder-only layers on generic paths first, and fine-tune it using product-centric single-hop paths for knowledge completion and user-centric multi-hop paths for path reasoning. Experiments on two public data sets (ML1M and LFM1M) show that KGGLM is more effective than state-of-the-art baselines under knowledge completion and recommendation.

2 Preliminaries

We start by introducing the concept of knowledge graph, a data structure used to store organized information about facts as a graph.

Definition 2.1: Knowledge Graph. *Given an entity set $\mathcal{V}$ and a relation set $\mathcal{R}$, a knowledge graph is denoted as $\mathcal{G} = \{(h, r, t) \mid h, t \in \mathcal{V}, r \in \mathcal{R}\}$. Each triplet (h, r, t) is a connection between the head entity h and tail entity t by the relation r. Note that $\mathcal{R}$ includes relations from canonical (r) and inverse (r^{-1}) directions.*

We assume there exists a subset of entities $\mathcal{I} \subset \mathcal{V}$ ($|\mathcal{I}| = N$) subject to recommendation (e.g., movies in the cinema domain or songs in the music domain). Due to the incomplete nature, knowledge completion along the recommendation pipeline involves predicting missing links between products $\mathcal{I}$ and, without loss of generalization, entities of other types ($\mathcal{V} \setminus \mathcal{I}$), expected to better inform the subsequent knowledge-aware modeling. Let $\mathcal{P}_\mathcal{G}$ denote a set of 1-hop paths (i.e., triplets) sampled from $\mathcal{G}$, with path $p \in \mathcal{P}_\mathcal{G}$ represented as (e_h, r, e_t) with entity $e_h = i \in \mathcal{I}$ and entity $e_t = e \in \mathcal{V} \setminus \mathcal{I}$. Hence, the knowledge completion task can be expressed as follows.

Definition 2.2: Knowledge Completion. *Given the path set $\mathcal{P}_\mathcal{G}$, knowledge completion aims to predict the tail entity $e_t \in \mathcal{V} \setminus \mathcal{I}$ in an incomplete triplet $(e_h, r, ?)$, such that $(e_h = i, r, e_t) \in \mathcal{P}_\mathcal{G}$. Formally, it means that $\hat{e_t} = \arg\max_{e \in \mathcal{V} \setminus \mathcal{I}} f_r(e_h, e)$ where $f_r(e_h, e)$ is a function scoring the plausibility of triplet (e_h, r, e).*

We then assume there exists a user set $\mathcal{U}$ ($|\mathcal{U}| = M$). Their relation s with products $\mathcal{I}$ (e.g., click or purchase) can be modelled as a user-product bipartite graph $\mathcal{H}$. To this end, we denote the binary interaction matrix as $\mathbf{Y} \in \{0, 1\}^{M \times N}$, where $y_{ui} = 1$ indicates that the user $u \in \mathcal{U}$ has interacted with product $i \in \mathcal{I}$, otherwise $y_{ui} = 0$. The user-product bipartite graph can be hence represented as $\mathcal{H} = \{(h, r, t) \mid h \in \mathcal{U}, t \in \mathcal{I}, r \in \{s, s^{-1}\}, y_{ht} = 1\}$. With this, we can combine the knowledge graph $\mathcal{G}$ with the user-item bipartite graph $\mathcal{H}$ into a collaborative knowledge graph $C = \mathcal{G} \cup \mathcal{H}$ for the path reasoning task. Let then $\mathcal{P}_C$ denote a set of n-hop paths sampled from C that connect users to products. Each path $p \in \mathcal{P}_C$ is in the form $(e_0, r_1, e_1, \ldots, r_n, e_n)$ with $(e_0 = u, s, e_n = i) \in \mathcal{H} \subset C$.

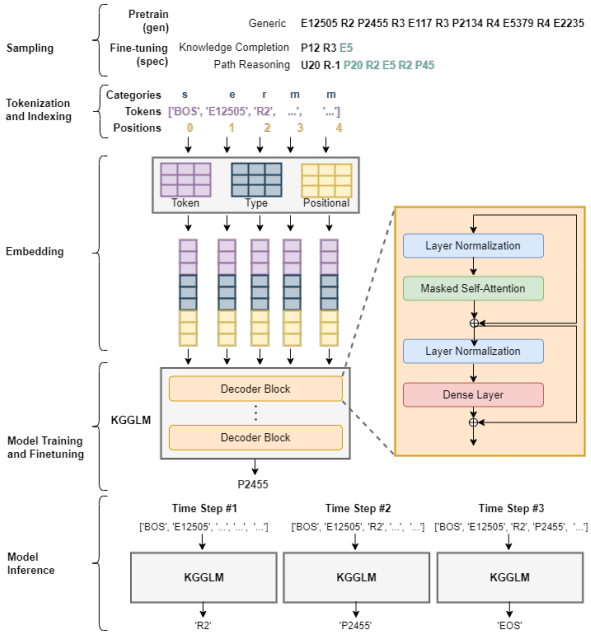

Figure 1: Methodology: We sample, tokenize, and encode generic paths from the collaborative knowledge graph. We train a model using Transformer decoder-only layers on such sequences, fine-tune on task-specific paths, and infer.

Definition 2.3: Path Reasoning. *Given the path set $\mathcal{P}_C$, path reasoning aims to predict the missing relations and entities r_{t+1}, e_{t+1} for $t = 0, \ldots n-1$ in an incomplete chain $(e_0 = u, ?)$, such that $(e_0 = u, r_1, e_1, \ldots, r_n, e_n = i) \in \mathcal{P}_C$. Formally, $(\hat{r_{t+1}}, \hat{e_{t+1}}) = \arg\max_{r \in \mathcal{R} \cup \{s, s^{-1}\}, e \in \mathcal{V} \cup \mathcal{U}} f_r(e_h, e)$ for $t = 0, 1, \ldots, n-1$. The function $f_r(e_h, e)$ scores the plausibility of each triplet (e_h, r, e).*

Our study in this paper aims to learn a function f parametrized by θ that accurately predicts both missing entities in triplets from $\mathcal{P}_\mathcal{G}$ as for the knowledge completion step and relevant paths in $\mathcal{P}_C$ as for path reasoning. Our objective can be expressed as follows.

Definition 2.4: Knowledge Representation Learning. *Given two sets of paths, $\mathcal{P}_\mathcal{G}$ from a knowledge graph $\mathcal{G}$ and $\mathcal{P}_C$ from a collaborative knowledge graph C, the goal is to learn a function f parameterized by θ that optimizes the following objective: $\tilde{\theta} = \arg\max_\theta \mathbb{E}_{p=(e_0,\ldots) \in \mathcal{P}_\mathcal{G} \cup \mathcal{P}_C} \sum_{t=0}^{|p|-1} f^\theta_{r_{t+1}}(e_t, e_{t+1})$.*

Note that our formalization and approach can be extended to other knowledge-related steps within the recommendation pipeline.

3 Methodology: KGGLM

We outline our methodology for generalizable knowledge representation learning relevant in recommendation pipelines (Fig. 1).

Sampling. Knowledge graphs are massive, comprising a multitude of entities and relations. To learn from them, we require large amounts of training sequences. To this end, we employ an off-the-shelf random walk algorithm [40] to extract n-hop paths from the collaborative knowledge graph C. Each path p is in the form $(e_0, r_1, e_1, \ldots, r_n, e_n)$ with $(e_t, r_{t+1}, e_{t+1}) \in C$ for $t = 0, \ldots, n-1$.

Differently from prior work [12], the sampled paths do not satisfy any requirements (i.e., head and tail entities can be of any type) to ensure that there is enough diversity for the model to then solve various downstream steps[1]. An example path with $n = 5$ can be `E12505 R2 P2455 R3 E11407 R3 P2134 R4 E5379 R4 P2235`. As a post-sampling step, we add `BOS` and `EOS` to the sequences, where `BOS` and `EOS` are special tokens indicating the start and end of a sequence. This step would lead to `['BOS', 'E12505', 'R2', 'P2455', 'R3', 'E11407', 'R3', 'P2134', 'R4', 'E5379', 'R4', 'P2235', 'EOS']`, as an example.

Tokenization and Indexing. Language models process sequences of tokens. In our case, the vocabulary of tokens is denoted by $\mathcal{D} = \mathcal{V} \cup \mathcal{U} \cup \mathcal{R} \cup \{s, s^{-1}\} \cup \{\texttt{BOS}, \texttt{EOS}\}$. To move from paths to tokenized sequences, we adopt a word-level tokenizer that preserves the integrity of entities and relations. The example path `BOS E12505 R2 P2455 R3 E11407 R3 P2134 R4 E5379 R4 P2235 EOS` would be split into `['BOS', 'E12505', 'R2', 'P2455', 'R3', 'E11407', 'R3', 'P2134', 'R4', 'E5379', 'R4', 'P2235', 'EOS']`. Tokenized sequences are then mapped to non-negative integers using an injective function $g : \mathcal{D} \to \mathbb{N}$. By applying g to each token in the example sequence, we would get the indexed sequence `[0, 2, 2455, 3, 11407, 3, 2134, 4, 5379, 4, 2235, 1]`.

Embedding. To capture their semantic meaning, each token is associated with a respective embedding. We define $\mathcal{E}_T : \mathbb{N} \to \mathbb{R}^d$ as a function that maps indexed tokens to continuous vectors of size d. We also introduce category embeddings, which differentiate between entities, relations, and special tokens. We define $\mathcal{E}_C : \mathbb{N} \to \mathbb{R}^d$ as a function that maps token categories, assumed to be indexed as well, to continuous vectors of size d. Finally we consider positional embeddings, which encode the relative positions of tokens within a path to maintain the sequence order. We define $\mathcal{E}_P : \mathbb{N} \to \mathbb{R}^d$ as a function that maps token positions in the sequence to continuous vectors of size d. Given the i-th token $\mathbf{t}_i$ of a training sequence, its embedding is computed by adding the corresponding embeddings $\mathcal{E}(\mathbf{t}_i) = \mathcal{E}_T(\mathbf{t}_i) + \mathcal{E}_C(\mathbf{t}_i) + \mathcal{E}_P(i)$, to learn meaningful representations.

Model Training. We adopt Transformer [23, 30] decoder-only layers parametrized by θ, with l layers and each layer with h heads in multi-head self-attention, to train an auto-regressive model. For an input sequence p_i at layer $i \in \{0, 1, \ldots, l-1\}$, the encoded sequence p_{i+1} is computed as $\text{FFN}_i\left(\text{Attention}(p_i W_Q, p_i W_K, p_i W_V)\right)$ where W_Q, W_K, and $W_V \in \mathbb{R}^{d \times d_h}$ are weight matrices to project the query, key, and value, respectively. Note that $d_h = d/h$ is the dimensionality of each attention head. The function FFN_i is a fully-connected layer with `ReLU` activation. The $\text{Attention}(Q, K, V)$ is defined as $\text{Softmax}(QK^\top/\sqrt{d_h})V$ where the scaling factor $\sqrt{d_h}$ maintains the order of magnitude in feature maps. We use a causal masking strategy to avoid any leakage of future information [32]. The probability of each token being the next in the sequence is computed as $P(t_k \mid t_1, \ldots, t_{k-1}) = \text{Softmax}(p_l W + b)$ if $t_1, \ldots, t_k \in \mathcal{P}_C$ else 0, with W and b being the weight matrices and bias vectors. This ensures that the predicted sequence exists in $\mathcal{C}$. The loss can be finally defined as $\mathcal{L} = \sum_{(t_1,\ldots,t_j)\in\mathcal{P}} \log P(t_j \mid t_1, \ldots, t_{j-1})$.

[1]Though our study focuses on knowledge completion and path reasoning, we seek a methodology seamlessly extensible to other knowledge-related steps in the pipeline.

Model Fine-tuning. The paths sampled for training did not meet any criteria, allowing the model to learn foundational representations of entity and relations. To specialize them, we perform a light fine-tuning phase. Specifically, under knowledge completion, fine-tuning involves sampling product-centric paths that connect products to other non-user entities. Each path p is represented as (e_h, r, e_t) where $e_h = i \in \mathcal{I}$ and $e_t = e \in \mathcal{V} \setminus \mathcal{I}$. For example, a path might be "`E12 R3 E5`", where `E12` is a product. In contrast, for path reasoning, fine-tuning involves sampling user-centric paths that connect users to products. Each path p is in the form $(e_0, r_1, e_1, \ldots, r_n, e_n)$ where $(e_0 = u, r_1, e_n = i) \in \mathcal{H} \subset \mathcal{C}$. An example path of this type might be "`U20 R-1 P20 R2 E5 R2 P45`". Note that the number of sampled paths for fine-tuning is smaller compared to the amount for training from scratch, ensuring efficiency.

Model Inference. We consider a prefix sequence such as `[BOS] E2185 R3` for knowledge completion and `[BOS] U881 R-1` for path reasoning. The trained (fine-tuned) model generates paths extending from this prefix, with a hop length of 1 and 3 for the two steps [35], respectively. Let the path prefix be denoted as $[\text{BOS}, t_1, \ldots, t_k]$ where $t_i \in \mathcal{D}$ are tokens. During decoding, the model generates candidate paths $\hat{p} = [\text{BOS}, t_1, \ldots, t_k, t_{k+1}, \ldots, t_m]$ by extending the prefix with other tokens. To promote diversity, we exclude candidate paths seen during training or already decoded. Each candidate path $\hat{p}$ is assigned a cumulative probability $P(\hat{p}) = \prod_{i=k+1}^{m} P(t_i \mid t_1, t_2, \ldots, t_{i-1})$ which reflects model confidence. Candidate paths are ranked based on their cumulative probabilities, and the top k unique (i.e., with distinct final tokens) paths $\hat{p}^{(1)}, \ldots, \hat{p}^{(k)}$ are chosen. Their final tokens are recommended products (tail entities) for recommendation (knowledge completion).

4 Experimental Evaluation

In this section, we aim to understand whether our approach, `KGGLM`, can surpass state-of-the-art baselines on real-world data sets under knowledge completion (**RQ1**) and recommendation (**RQ2**) steps.

4.1 Experimental Setup

We detail the data sets used, introduce the evaluation metrics employed, and the baseline models chosen for comparison.

Data Preparation. We conducted experiments on ML1M [13] and LFM1M [27] (Tab. 1). For ML1M, we used the knowledge graph from [7], and for LFM1M, the knowledge graph from [33]. For both, we removed products not present in the knowledge graph (and their interactions), as in [3]. We also excluded products and users with fewer than five interactions to control sparsity. We conducted a random split of the knowledge graph's triplets (excluding user-product triplets), allocating 60% for training, 20% for validation,

Table 1: Data Sets: ML1M has a higher density and average product degree compared to LFM1M, which has more products and interactions. ML1M features more entity types and relation types, while LFM1M has a larger knowledge graph.

Collaborative Components			Knowledge Components		
Characteristic	**ML1M**	**LFM1M**	**Characteristic**	**ML1M**	**LFM1M**
Users	6,040	4,817	Entities (Types)	13,804 (12)	17,492 (5)
Products	2,984	12,492	Relations (Types)	193,089 (11)	219,084 (4)
Interactions	932,295	1,091,275	Avg. Degree (All)	28.07	25.05
Density	0.05	0.01	Avg. Degree (Products)	64.86	17.53

Table 2: [RQ1] Knowledge Completion: KGGLM (gen+spec) beats all the baseline methods on ML1M, while TorusE and ComplEx lead to the best performance on LFM1M.

Knowledge Modeling Method	ML1M		LFM1M	
	MRR ↑	HITS@1 ↑	MRR ↑	HITS@1 ↑
TransE [6]	0.19	0.14	0.76	0.66
TransH [34]	0.19	0.15	0.76	0.66
TransD [15]	0.16	0.07	0.31	0.01
TransR [16]	0.13	0.06	0.35	0.19
TorusE [11]	0.20	0.15	**0.79**	**0.69**
RotatE [28]	0.22	0.13	0.07	0.02
RESCAL [20]	0.20	0.16	0.29	0.21
DistMult [37]	0.19	0.11	0.17	0.07
ComplEx [31]	0.23	0.15	0.68	0.58
TuckER [2]	0.10	0.04	0.76	0.66
Analogy [17]	0.20	0.13	0.50	0.37
HolE [19]	0.14	0.10	0.33	0.23
ConvE [10]	0.10	0.04	0.76	0.67
KGGLM (gen-only)	0.38	0.29	0.26	0.20
KGGLM (gen+spec)	**0.39**	**0.30**	0.09	0.04

and 20% for testing. These triplets were used for knowledge completion experiments. We then applied a temporal per-user split on user-product interaction triplets (using the oldest 60% for training, 20% for validation, and the most recent 20% for testing). These user-product triplet sets, combined with the respective knowledge graph's triplets, were used for recommendation experiments.

Metrics Computation. For knowledge completion, we evaluate performance by measuring the Mean Reciprocal Rank (MRR), i.e., the average reciprocal rank of the first correct prediction, and the Hits (H@k = 1), i.e., the proportion of cases where the target entity appears at the first position. For recommendation, we again use MRR for between-step comparison, together with the Normalized Discounted Cumulative Gain (NDCG), i.e., the quality of the recommended list given ranking positions of relevant products.

Baselines Choice. We compare with four types of method[2]. General-purpose methods include RESCAL [20], DistMult [37], HolE [19], ComplEx [31], Analogy [17], TransE [6], TransH [34], TransR [16], TransD [15], TorusE [11], RotatE [28], ConvE [10], and TuckER [2]. Special-purpose methods for path reasoning cover PGPR [35], CAFE [36], UCPR [29], and PLM-Rec [12]. We also include special-purpose, knowledge-aware recommendation methods that do not use path reasoning (CKE [39], CFKG [1], and KGAT [33]) and traditional recommendation methods (MostPopular and BPRMF [24]).

Implementation Details. We consider DistilGPT-2 [26] (l = 6, h = 12) as the model, trained with the Adam optimizer on an Nvidia RTX A6000 GPU. The embedding size d is set to 768. For training (fine-tuning), the path hop n is set to 5 (1 for knowledge completion, while 3 for path reasoning), the sampled paths are 500 (250), and we use a learning rate of 0.0002, a batch size of 256 for both the training and fine-tuning, and a number of training epochs of 3 (2).

4.2 Experimental Results

We provide experimental results to answer the research questions.

RQ1: Effectiveness on Knowledge Completion. In a first analysis, we assessed the performance of our approach, KGGLM, against general-purpose baselines under the knowledge completion task. Tab. 2 reports the MRR and HITS@1 scores across the ML1M and LFM1M data sets. This comparison aims to show the relative strengths and weaknesses of our approach in different domains.

[2]The grid and best hyperparameters for each baseline are provided in the repository.

Table 3: [RQ2] Recommendation: KGGLM (gen+spec) consistently outperforms all the baseline methods on both data sets, achieving the highest scores in both MRR and NDCG.

Knowledge Modeling Method	ML1M		LFM1M	
	MRR ↑	NDCG@10 ↑	MRR ↑	NDCG@10 ↑
TransE [6]	0.23	0.28	0.10	0.12
TransH [34]	0.20	0.26	0.14	0.18
TransD [15]	0.18	0.23	0.13	0.17
TransR [16]	0.12	0.16	0.11	0.15
TorusE [11]	0.21	0.26	0.14	0.18
RotatE [28]	0.15	0.20	0.20	0.25
RESCAL [20]	0.20	0.26	0.19	0.24
DistMult [37]	0.22	0.28	0.25	0.30
ComplEx [31]	0.21	0.26	0.23	0.28
TuckER [2]	0.22	0.28	0.10	0.12
Analogy [17]	0.21	0.26	0.28	0.33
HolE [19]	0.19	0.24	0.14	0.19
ConvE [10]	0.22	0.27	0.11	0.13
MostPopular	0.22	0.26	0.10	0.12
BPRMF [24]	0.23	0.29	0.06	0.08
CKE [39]	0.23	0.30	0.27	0.33
CFKG [1]	0.21	0.27	0.10	0.13
KGAT [33]	0.24	0.31	0.24	0.30
PGPR [35]	0.21	0.28	0.14	0.18
UCPR [29]	0.19	0.26	0.26	0.32
CAFE [36]	0.15	0.21	0.09	0.14
PLM-Rec [12]	0.18	0.27	0.19	0.28
KGGLM (gen-only)	0.07	0.11	0.20	0.27
KGGLM (gen+spec)	**0.31**	**0.41**	**0.45**	**0.53**

Our results reveal that KGGLM (gen+spec) excels on the ML1M data set, where it achieves the highest MRR (0.39) and HITS@1 (0.30) scores. This superior performance suggests that combining generic and specific paths aids the model's ability to accurately rank entities, although the gain over using only generic paths with KGGLM (gen-only) is not substantial. On the LFM1M data set, methods like TorusE and ComplEx show better results than KGGLM variants. TorusE stands out with the highest MRR (0.79) and HITS@1 (0.69) scores. This may be due to the lower average degree of products in LFM1M (Tab. 1) and the need for more paths than what used for training and tuning KGGLM variants to model such representations.

Answer to RQ1. KGGLM (gen+spec) *shows benefits on ML1M but not LFM1M for knowledge completion. Fine-tuning has a minimal yet positive impact only on ML1M. Diverging findings suggest that effectiveness depends on domain-specific knowledge factors.*

RQ2: Effectiveness on Recommendation. In a second stage, we evaluated the performance of the various methods against our approach, KGGLM, with a focus on recommendation. Specifically, Tab. 3 reports the MRR and NDCG@10 for both general- and special-purpose methods on the ML1M and LFM1M data sets.

Observations from the table reveal that KGGLM (gen+spec) outperforms all other methods in both MRR and NDCG@10 across both data sets. Specifically, it achieves the highest scores of 0.31 (29.17% gain with respect to the best baseline KGAT) in MRR and 0.41 (32.26% of gain against KGAT) in NDCG@10 on ML1M, and 0.45 (60.71% of gain against Analogy) in MRR and 0.53 (60.61% of gain against Analogy and CKE) in NDCG@10 on LFM1M. This consistently superior performance suggests that the combination of generic and specific knowledge is essential. The KGGLM (gen-only) variant,

which uses only general knowledge, showed lower effectiveness, even when compared to general-purpose baselines.

Answer to RQ2. `KGGLM (gen+spec)` *shows a clear advantage over both generic and specialized state-of-the-art methods. Fine-tuning leads to substantial gains.*

5 Conclusions and Future Works

We introduced a novel approach that leverages language models as a foundational framework for knowledge representation learning, with generalizable applicability to steps along the recommendation pipeline. Our approach involves a decoder-only model training strategy that incorporates a training on generic paths to understand the overall structure of the knowledge graph, followed by fine-tuning with paths specific of the downstream step. This dual-phase strategy has shown improvements over baselines, highlighting its effectiveness in enhancing task-specific metrics. While we acknowledge that our results are not yet comprehensive enough to be published as a full research paper, we expect that our late-breaking findings can influence a large audience working in either knowledge completion, path reasoning, or more in general modern recommendation.

In the next steps, we plan to consolidate our findings by exploring the use of larger model architectures, such as GPT-3. We will experiment with large and more diverse sampled path sets, incorporating a mixture of path types and lengths, to enhance robustness. Additionally, we aim to address the limitations of random walk data creation, which diminishes the representation of second-order statistics in the training data. Validation on a variety of data sets will be conducted to ensure generalizability across domains, including ablation studies on the approach's parameters and the required computational loads. Finally, considering that optimal performance, particularly in recommendation tasks, necessitates fine-tuning the model and consequently results in two distinct sets of weights, we will investigate the extent to which these two resulting models align and whether their representations reside within the same or a similar continuous space, for completeness.

Acknowledgments. We acknowledge financial support under the National Recovery and Resilience Plan (NRRP), Miss. 4 Comp. 2 Inv. 1.5 - Call for tender N. 3277 published on Dec 30, 2021 by the Italian Ministry of University and Research (MUR) funded by the European Union – NextGenerationEU. Prj. Code ECS0000038 eINS Ecosystem of Innovation for Next Generation Sardinia, CUP F53C22000430001, Grant Assignment Decree N. 1056, Jun 23, 2022 by the MUR.

References

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* 11, 9 (2018), 137.

[2] Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *Proc. of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*. ACM.

[3] Giacomo Balloccu, Ludovico Boratto, Christian Cancedda, Gianni Fenu, and Mirko Marras. 2023. Knowledge is Power, Understanding is Impact: Utility and Beyond Goals, Explanation Quality, and Fairness in Path Reasoning Recommendation. In *Proc. of the 45th European Conference on Information Retrieval (ECIR 2023)*, Vol. 13982. Springer, 3–19.

[4] Giacomo Balloccu, Ludovico Boratto, Gianni Fenu, and Mirko Marras. 2022. Post Processing Recommender Systems with Knowledge Graphs for Recency, Popularity, and Diversity of Explanations. In *Proc. of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2022*. ACM, 646–656.

[5] Giacomo Balloccu, Ludovico Boratto, Gianni Fenu, and Mirko Marras. 2023. Reinforcement recommendation reasoning through knowledge graphs for explanation path quality. *Knowl. Based Syst.* 260 (2023), 110098.

[6] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Proc. of the 27th Annual Conference on Neural Information Processing Systems (NIPS 2013)*. Curran Associates, Inc., 2787–2795.

[7] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *Proc. of the The World Wide Web Conference (WWW 2019)*. ACM.

[8] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. 2020. Knowledge Graph Completion: A Review. *IEEE Access* 8 (2020), 192435–192456.

[9] Danilo Dessì, Mauro Dragoni, Gianni Fenu, Mirko Marras, and Diego Reforgiato Recupero. 2019. Evaluating neural word embeddings created from online course reviews for sentiment analysis. In *Proc. of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019*. ACM, 2124–2127.

[10] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In *Proc. of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*. AAAI Press, 1811–1818.

[11] Takuma Ebisu and Ryutaro Ichise. 2018. TorusE: Knowledge Graph Embedding on a Lie Group. In *Proc. of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 1819–1826.

[12] 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 *Proc. of the ACM Web Conference 2022 WWW 2022)*. ACM, 946–955.

[13] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 19 (2015), 19.

[14] Subhi Issa, Onaopepo Adekunle, Fayçal Hamdi, Samira Si-Said Cherfi, Michel Dumontier, and Amrapali Zaveri. 2021. Knowledge Graph Completeness: A Systematic Literature Review. *IEEE Access* 9 (2021), 31322–31339.

[15] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge Graph Embedding via Dynamic Mapping Matrix. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*. ACM, 687–696.

[16] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proc. of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015) (AAAI'15)*. AAAI Press, 2181–2187.

[17] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical Inference for Multi-relational Embeddings. In *Proc. of the 34th International Conference on Machine Learning (ICML 2017)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, 2168–2178.

[18] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In *Proc. of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*. ACL, 1–8.

[19] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic Embeddings of Knowledge Graphs. In *Proc. of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*. AAAI Press, 1955–1961.

[20] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proc. of the 28th International Conference on International Conference on Machine Learning (ICML 2011) (ICML'11)*. Omnipress, Madison, WI, USA, 809–816.

[21] Thamizhiniyan Pugazhenthi and Huizhi Liang. 2022. Improving Conversational Recommender Systems via Knowledge Graph-based Semantic Fusion with Historical Interaction Data. In *Proc. of the IEEE International Conference on Big Data, Big Data 2022*. IEEE, 4303–4312.

[22] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[23] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog* 1, 8 (2019), 9.

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proc. of the 25th Conference on Uncertainty in Artificial Intelligence (UAI 2009)*. AUAI Press, 452–461.

[25] Amina Samih, Abderrahim Ghadi, and Abdelhadi Fennan. 2020. *Translational-randomwalk embeddings-based recommender systems: a pragmatic survey.* 957–966.

[26] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019). arXiv:1910.01108

[27] Markus Schedl. 2016. The LFM-1b Dataset for Music Retrieval and Recommendation. In *Proc. of the 2016 ACM on International Conference on Multimedia Retrieval (ICMR 2016)*. ACM, 103–110.

[28] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* (2019).

[29] Chang-You Tai, Liang-Ying Huang, Chien-Kun Huang, and Lun-Wei Ku. 2021. User-Centric Path Reasoning towards Explainable Recommendation. In *Proc. of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*. Association for Computing Machinery, 879–889.

[30] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023).

[31] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. [n. d.]. Complex Embeddings for Simple Link Prediction. In *Proc. of the 33nd International Conference on Machine Learning (ICML 2016), volume = 48, pages = 2071–2080, publisher = JMLR.org, year = 2016, timestamp = Wed, 29 May 2019 08:41:46 +0200, biburl = https://dblp.org/rec/conf/icml/TrouillonWRGB16.bib, bibsource = dblp computer science bibliography, https://dblp.org*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.).

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proc. of the Annual Conference on Neural Information Processing Systems (NIPS 2017)*. 5998–6008.

[33] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proc. of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (SIGKDD 2019)*.

[34] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. *Proc. of the AAAI Conference on Artificial Intelligence (AAAI 2014)* 28, 1 (2014), 1–8.

[35] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *Proc. of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*. 285–294.

[36] Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard de Melo, S. Muthukrishnan, and Yongfeng Zhang. 2020. CAFE: Coarse-to-Fine Neural Symbolic Reasoning for Explainable Recommendation. In *Proc. of the 29th ACM International Conference on Information & Knowledge Management (CIKM 2020)*.

[37] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Prco. of the 3rd International Conference on Learning Representations (ICLR 2015)*, Yoshua Bengio and Yann LeCun (Eds.).

[38] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge Graph Contrastive Learning for Recommendation. In *Proc. of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2022*. ACM, 1434–1443.

[39] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2016)*. 353–362.

[40] Jiajun Zhou, Bo Liu, Lu Deng, Yaofeng Chen, and Zhefeng Xiao. 2018. On Random Walk Based Weighted Graph Sampling. *IEICE Trans. Inf. Syst.* 101-D, 2 (2018), 535–538.