
Unlearning of Knowledge Graph Embedding via Preference Optimization

Jiajun Liu¹, Wenjun Ke^{1,2*}, Peng Wang^{1,2*}, Yao He³, Ziyu Shang¹, Guozheng Li¹, Zijie Xu¹, and Ke Ji⁴

¹School of Computer Science and Engineering, Southeast University

²Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education

³School of Institute of Collaborate Innovation, University of Macau

⁴The Chinese University of Hongkong, Shenzhen

{jiajliu, kewenjun, pwang, ziyus1999, gzli, zijiexu}@seu.edu.cn, mc46477@um.edu.mo,
keji@link.cuhk.edu.cn

Abstract

Existing knowledge graphs (KGs) inevitably contain outdated or erroneous knowledge that needs to be removed from knowledge graph embedding (KGE) models. To address this challenge, knowledge unlearning can be applied to eliminate specific information while preserving the integrity of the remaining knowledge in KGs. Existing unlearning methods can generally be categorized into exact unlearning and approximate unlearning. However, exact unlearning requires high training costs while approximate unlearning faces two issues when applied to KGs due to the inherent connectivity of triples: (1) It fails to fully remove targeted information, as forgetting triples can still be inferred from remaining ones. (2) It focuses on local data for specific removal, which weakens the remaining knowledge in the forgetting boundary. To address these issues, we propose GraphDPO, a novel approximate unlearning framework based on direct preference optimization (DPO). Firstly, to effectively remove forgetting triples, we reframe unlearning as a preference optimization problem, where the model is trained by DPO to prefer reconstructed alternatives over the original forgetting triples. This formulation penalizes reliance on forgettable knowledge, mitigating incomplete forgetting caused by KG connectivity. Moreover, we introduce an out-boundary sampling strategy to construct preference pairs with minimal semantic overlap, weakening the connection between forgetting and retained knowledge. Secondly, to preserve boundary knowledge, we introduce a boundary recall mechanism that replays and distills relevant information both within and across time steps. We construct eight unlearning datasets across four popular KGs with varying unlearning rates. Experiments show that GraphDPO outperforms state-of-the-art baselines by up to 10.1% in MRR_{Avg} and 14.0% in MRR_{F1} . Further analysis confirms that GraphDPO more effectively removes target knowledge while preserving surrounding context.

1 Introduction

Knowledge Graph Embedding (KGE) [1, 2] aims to embed entities and relations in knowledge graphs (KGs) [3] into low-dimensional vectors, which is a crucial method for many knowledge-driven applications, such as question answering [4], semantic search [5], and information retrieval for large language models (LLMs) [6]. However, training data in KGs may contain incorrect and outdated

*Corresponding author.

information that needs to be removed [7]. For instance, nearly 5% of the total knowledge in YAGO [8] KG is incorrect [9]. Besides, vast outdated knowledge becomes irrelevant or incorrect due to changes in the real world [10]. For example, Wikidata KG has been edited more than 2 billion times from 2012 to 2024 , which means up to 2 billion outdated pieces of knowledge should be forgotten.

One promising approach to address these challenges is knowledge unlearning, which involves deleting specific knowledge from models [11]. However, existing knowledge unlearning methods are primarily designed for LLMs [12] and computer vision models (CVMs) [13], with limited attention paid to unlearning in KGE models. Furthermore, existing KGE methods that leverage LLMs [10] mainly focus on the addition and modification of knowledge, overlooking the knowledge unlearning in KGs. Additionally, these LLM-based approaches tend to incur significant computational costs and require large-scale retraining, limiting their scalability and efficiency. In this paper, we study the Unlearning of Knowledge Graph Embedding (UKGE) task, which aims to forget incorrect or outdated knowledge in pre-trained KGE models. Compared with methods based on LLMs, UKGE provides a more lightweight alternative without LLMs in scenarios where computational resources are limited. The crucial factor for the success of UKGE lies in effectively removing unlearned knowledge while preserving the remaining knowledge. As illustrated in Figure 1, while the triple (*James Gordon, friend, Selina Kyle*) needs to be deleted, other triples such as (*James Gordon, friend, Bruce Wayne*) should remain. Existing unlearning methods can be generally categorized into exact unlearning [14] and approximate unlearning [15]. Exact unlearning methods retrain the entire dataset to forget the specific knowledge [16], thus coming with huge training costs.

To mitigate this drawback, approximate unlearning [17] updates parameters using only forgetting data, thereby reducing training expenses. While approximate unlearning has made significant progress, it still suffers from the following issues when directly applied to UKGE due to the interconnected nature of KGs. Firstly, it fails to fully remove forgetting knowledge, as forgetting triples can still be inferred from the remaining ones. Secondly, it focuses on local data for specific removal, which weakens remaining knowledge in the forgetting boundary.

To address the above issues, we reframe unlearning as a preference optimization problem and propose a novel unlearning framework for knowledge **Graph** embedding with **DPO**, named **GraphDPO**, which efficiently forgets unwanted knowledge and preserves remaining knowledge. Specifically, the optimization objective for negative samples through preference optimization aligns with the objective of unlearning. Additionally, it allows for efficient sampling of positive samples on the graph structure by leveraging the graph connectivity features. Therefore, the preference optimization method is well-suited for addressing the challenge of forgetting learning on graph structures. Firstly, to mitigate the incomplete forgetting caused by the structural connectivity in KGs, preference optimization is modeled in such a way that forgetting triples are assigned larger penalty scores, thus allowing forgetting knowledge to be efficiently forgotten. Moreover, we design an out-boundary sampling strategy to construct alternative triples that are structurally distant from the forgetting triples, thus reducing their semantic and relational overlap with the retained knowledge. Then the DPO algorithm trains the model to prefer reconstructed alternatives over the original forgetting triples, suppressing their influence and preventing indirect reinforcement through retained structures. Secondly, to tackle the integrity issue of remaining knowledge in forgetting boundary, GraphDPO introduces a boundary recall mechanism to ensure minimal destruction of remaining knowledge. In detail, we replay and distill remaining knowledge at the forgetting boundary to preserve it within a single time step and across multiple time steps, respectively.

To validate the effectiveness of our method, we reorganize four commonly used datasets, FB15K-237 [18], WN18RR [19], CoDEx-L [20], and Yago3-10 [21], and construct eight unlearning datasets with 10% and 20% unlearning rates, specifically for the novel UKGE task. Experimental results show that GraphDPO achieves optimal results in MRR_{Avg} and MRR_{F1} on most datasets compared to other strong approximate baselines. Our main contributions can be divided as follows:

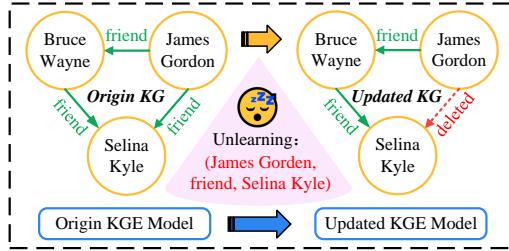


Figure 1: Illustration of unlearning in knowledge graph embedding (UKGE). The forgetting knowledge is (*James Gordon, friend, Selina Kyle*), while other knowledge retains in the updated KG.

- We propose GraphDPO, a novel UKGE framework that reformulates unlearning as preference optimization. By combining DPO with a graph-aware out-boundary sampling strategy, GraphDPO effectively reduces inference leakage from remaining knowledge.
- We design a boundary recall mechanism that explicitly preserves knowledge near the forgetting boundary. It combines knowledge replay and distillation to retain relevant context during the forgetting process, ensuring minimal degradation of remaining knowledge.
- We construct and release eight benchmark datasets for UKGE across four standard KGs with varying unlearning rates. Extensive experiments show that GraphDPO consistently outperforms strong approximate unlearning baselines, achieving gains of up to 10.1% in MRR_{Avg} and 14.0% in MRR_{F1} , while maintaining high training efficiency.

2 Related Work

Knowledge Unlearning. Knowledge unlearning aims to remove incorrect and outdated knowledge from machine learning models [13]. Existing unlearning methods can be categorized into exact and approximate approaches. Exact unlearning methods [14, 22] require re-training the model using all reserved data, resulting in significant training costs. To mitigate this issue, approximate unlearning methods [15, 17] update models using only the forgetting datasets, with little or no use of the reserved datasets, thereby reducing training time [23, 24]. Recent works utilize schema [25] or meta-learning [26] to forget knowledge in KGE models, however, they struggle to effectively forget while preserving remaining knowledge [27, 28, 29] due to the inherent connectivity of KGs.

Preference Optimization. Preference optimization seeks to align LLMs with human preferences and values [30], and can be categorized into online and offline methods [31]. Online algorithms incorporate reinforcement learning with supervised fine-tuning and policy optimization, which are inherently challenging to optimize [32]. To solve these issues, offline algorithms like DPO [33] directly compare different decision sequences to optimize models, resulting in more efficient performance [34]. As alignment techniques leverage both positive and negative samples, offline algorithms are well-suited for selectively forgetting and remaining knowledge. In this paper, we demonstrate the effectiveness of applying DPO for unlearning in KGs. More detailed related work list in Appendix A.

3 Methodology

3.1 Preliminary and Problem Statement

Knowledge Graph. A knowledge graph (KG) is denoted as $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, where \mathcal{E} , \mathcal{R} , and \mathcal{T} denote the set of entities, relations, and triples, respectively. A triple can be denoted as (h, r, t) , where $h \in \mathcal{E}$, $r \in \mathcal{R}$, and $t \in \mathcal{E}$ denote the head entity, the relation and the tail entity, respectively.

Knowledge Graph Embedding. Knowledge graph embedding (KGE) embeds the entities and relations in KGs into low-dimensional vector space \mathbb{R}^d , where d defines the embedding dimension. Specifically, KGE embeds the h , r , and t in each triple to $\mathbf{h} \in \mathbb{R}^d$, $\mathbf{r} \in \mathbb{R}^d$, and $\mathbf{t} \in \mathbb{R}^d$, respectively. A KGE model can be defined as $\mathcal{M} = \{\mathbf{E}, \mathbf{R}, f\}$, where \mathbf{E} , \mathbf{R} denote the set of embeddings of entities and relations, respectively, and f denotes the score function for triples. The optimization objective for KGE is to minimize $\mathbb{E}_{(h, r, t) \sim \mathcal{T}, (h', r', t') \sim \overline{\mathcal{T}}} [f(h', r', t') - f(h, r, t)]$, where $\overline{\mathcal{T}}$ denotes the set of negative triples.

Unlearning in Knowledge Graph Embedding. The set of the training dataset for a KG \mathcal{G} can be defined as $\mathcal{D}_t = \{(h^{(i)}, r^{(i)}, t^{(i)})\}_{i=1}^{N_T}$, where N_T denotes the number of training triples and $\mathcal{D}_t \subseteq \mathcal{T}$. The set of the forgetting dataset can be defined as $\mathcal{D}_f \subset \mathcal{D}_t$, and the set of the remaining dataset can be defined as $\mathcal{D}_r = \mathcal{D}_t \setminus \mathcal{D}_f$. For a pre-trained KGE model \mathcal{M}_{pre} , unlearning in knowledge graph embedding (UKGE) aims to update \mathbf{E} and \mathbf{R} in \mathcal{M}_{pre} to remove knowledge in \mathcal{D}_f and preserve the knowledge in \mathcal{D}_r . Specifically, the optimization objective of UKGE is to minimize $\mathbb{E}_{(h, r, t) \sim \mathcal{D}_r, (h', r', t') \sim \mathcal{D}_f} [f(h', r', t') - f(h, r, t)]$.

Preference Optimization. Following [33], we formulate preference optimization as follows. Given a model \mathcal{M}^{sft} trained by supervised fine-tuning, the answer pair $(y_1, y_2) \sim \mathcal{M}^{sft}(\cdot | x)$ can be

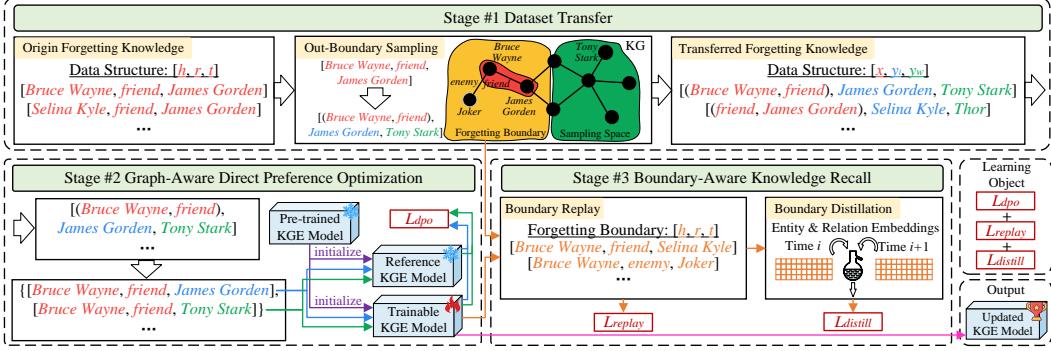


Figure 2: An overview of GraphDPO framework.

generated by \mathcal{M}^{sft} with each input x . The answer pair (y_1, y_2) will be labeled as (y_w, y_l) by human labels or reference policy model \mathcal{M}^{ref} , where y_w and y_l denote preferred and dis-preferred completion amongst (y_1, y_2) , respectively. For the comparison dataset $\mathcal{D}_c = \{(x^{(i)}, y_w^{(i)}, y_l^{(i)})\}_{i=1}^N$, where N denotes the number of \mathcal{D}_c , preference optimization aims to increase the output probability of $y_w^{(i)}$ and decrease the output probability of $y_l^{(i)}$. The objective of preference optimization is to minimize $\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}_c} [r_\phi(x, y_l) - r_\phi(x, y_w)]$, where r_ϕ denotes the training model initialized by \mathcal{M}^{sft} .

3.2 Framework

The framework of GraphDPO is illustrated in Figure 2. In Stage 1, we transfer the original forgetting knowledge using out-boundary sampling to datasets that are adaptive to preference optimization. In Stage 2, we apply the graph-aware direct preference optimization algorithm to achieve forgetting. In Stage 3, we design a boundary-aware knowledge recall method to retain the boundary knowledge.

3.3 Graph-Aware Direct Preference Optimization

To address the issue of forgotten knowledge being inferred from remaining knowledge, we reformulate unlearning as a preference optimization problem and employ a graph-aware direct preference optimization (DPO) algorithm with an out-boundary sampling strategy.

Dataset Transfer. To transfer unlearning in KGs to preference optimization, we extract the input x , the dis-preferred output y_l and construct the preferred output y_w required by the preference dataset from the forgetting dataset \mathcal{D}_f . Firstly, we extract each input x and dis-preferred output y_l from each sample of \mathcal{D}_f . Given the forgetting dataset $\mathcal{D}_f = \{(h^{(i)}, r^{(i)}, t^{(i)})\}_{i=1}^{N_f}$, where N_f denotes the number of \mathcal{D}_f , we denote \mathcal{D}_f^{po} from \mathcal{D}_f as follows:

$$\mathcal{D}_f^{po} = \{(h^{(i)}, r^{(i)}, t^{(i)}, s^{(i)})\}_{i=1}^{N_f}, s^{(i)} \in \{1, 0\} \quad (1)$$

where $(h^{(i)}, r^{(i)}, t^{(i)}) \in \mathcal{D}_f$, and $s^{(i)}$ is sampled from a Bernoulli distribution with $p(s) \sim Bernoulli(0.5)$. For each sample, when $s = 1$, we take the head entity h as the dis-preferred output y_l inferred by the input $x = (r, t)$. Similarly, when $s = 0$, we take the tail entity t as the dis-preferred output y_l inferred by the input $x = (h, r)$.

Secondly, we construct the preferred output y_w for each forgetting sample (x, y_l) . We randomly sample an entity $e \in \mathcal{E}$ such that $e \neq y_l$ from the knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$ as a preferred output answer y_w to the input x . The distribution of y_w can be formally approximated as follows:

$$p(y_w) \cong p(y_w|x, y_l) = \frac{1 - p(y_l)}{|\mathcal{E}| - 1}, \exists y_w \in \mathcal{E} \wedge y_w \neq y_l \quad (2)$$

Thus, the final forgetting dataset \mathcal{D}_f^{po} is defined as follows:

$$\mathcal{D}_f^{po} = \{(x^{(i)}, y_l^{(i)}, y_w^{(i)})\}_{i=1}^{N_f} \quad (3)$$

Meanwhile, we note that if $x = (h, r)$, then (x, y_l) and (x, y_w) correspond to the triples (h, r, y_l) and (h, r, y_w) , respectively. Similarly, if $x = (r, t)$, then (x, y_l) and (x, y_w) correspond to the triples

(y_l, r, t) and (y_w, r, t) , respectively. Finally, we transfer the forgetting dataset \mathcal{D}_f to \mathcal{D}_f^{po} . As shown in Figure 2 Stage 1, we transfer the forgetting knowledge from *[Bruce Wayne, friend, James Gorden]* to *[(Bruce Wayne, friend), James Gorden, Tony Stark]*.

Task Transfer. In this section, we reformulate the unlearning task defined over the forgetting set \mathcal{D}_f as a preference optimization problem over a constructed dataset \mathcal{D}_f^{po} . To support this formulation, we show that the two objectives are approximately equivalent to a linear transformation, thereby justifying the validity of the task transfer. We propose the following theorem:

Theorem 1 (Equivalence of Optimization Objectives) *Let E_u be the expectation of the unlearning objective with respect to \mathcal{D}_f , and E_p be the expectation of the preference objective with respect to \mathcal{D}_f^{po} , then we have:*

$$E_p = c_1 \cdot E_u - c_2 \quad (4)$$

where $c_1 > 1$ and $c_2 > 0$ are constants dependent on the candidate entity set size of \mathcal{D}_f .

See Appendix B for a detailed proof. Theorem 1 demonstrates that minimizing the unlearning expectation E_p is approximately equivalent to minimizing the preference expectation E_u . Thus, preference optimization serves as a principled surrogate for the unlearning task. Although E_u and E_p are not exactly equivalent in a strict distributional sense, their optimization objectives align closely, justifying that minimizing E_p effectively minimizes E_u .

Direct Preference Optimization. After reformulating the unlearning task as a preference optimization problem, we apply preference optimization algorithms to address it. Inspired by DPO [33], we use the DPO algorithm to minimize the expectation of the preference objective E_p . Specifically, given the pre-trained KGE model \mathcal{M} , and the transferred forgetting dataset \mathcal{D}_f^{po} , we initialize the reference policy model \mathcal{M}^{ref} using the parameters of \mathcal{M} . The loss for DPO is defined as follows, aiming to maximize $f_\theta(x, y_w)$ and minimize $f_\theta(x, y_l)$:

$$\mathcal{L}_{dpo} = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}_f^{po}} [\log \sigma(\beta \log \frac{f_\theta(x, y_w)}{f_{ref}(x, y_w)} - \beta \log \frac{f_\theta(x, y_l)}{f_{ref}(x, y_l)})] \quad (5)$$

where $f_\theta(\cdot)$ and $f_{ref}(\cdot) \in (0, 1)$ denote the score functions of \mathcal{M} and \mathcal{M}^{ref} , respectively. The hyperparameter β balances the preference and dis-preference data, and $\sigma(\cdot)$ represents the softmax function. We use TransE [27] as the base KGE model, where the score function is defined as $f(h, r, t) = Sigmoid(-|\mathbf{h} + \mathbf{r} - \mathbf{t}|_{L_2}) \in (0, 1)$, with \mathbf{h} , \mathbf{r} , and \mathbf{t} denoting the embeddings of h , r , and t , respectively. Notice that the loss of DPO algorithm is equal to minimizing the preference optimization object E_p [33], we apply the reference optimization in practice. During training, the parameters of \mathcal{M}^{ref} are frozen, and only the parameters of \mathcal{M} are updated. This approach allows the model \mathcal{M} to be optimized using the DPO algorithm. As shown in Figure 2 Stage 2, we use *[Bruce Wayne, friend, Tony Stark]* as the preferred triple, and *[Bruce Wayne, friend, James Gorden]* as the dis-preferred triple, which is fed into the reference model \mathcal{M}^{ref} and trainable model \mathcal{M} .

Out-Boundary Sampling. To enhance the preference-based unlearning, we propose an out-boundary sampling strategy to improve the selection of y_w . Specifically, we avoid sampling y_w from the close neighborhood of y_l , as neighboring entities tend to share similar embeddings [35], leading to less discriminative preference signals.

We first we denote forgetting boundary \mathcal{B}_{y_l} of y_l , and the boundary entities \mathcal{E}_{y_l} as follows:

$$\mathcal{B}_{y_l} = \{(h, r, t) \mid h = y_l \vee t = y_l\}, \quad \mathcal{E}_{y_l} = \{e \mid (e, r, t) \in \mathcal{B}_{y_l} \vee (h, r, e) \in \mathcal{B}_{y_l}\}. \quad (6)$$

Since single-hop neighboring entities tend to have similar entity representations in KGEs [35], if y_w is drawn from \mathcal{E}_{y_l} , then $f_\theta(x, y_w)$ in Equation 5 tends to approximate $f_\theta(x, y_l)$, making them hard to differentiate and optimize. Therefore, we modify the distribution of y_w as:

$$p_o(y_w) \cong p_o(y_w|x, y_l) = \frac{1 - p(y_l) - p(\mathcal{E}_{y_l})}{|\mathcal{E}| - 1 - |\mathcal{E}_{y_l}|} \approx \frac{1 - p(y_l) - p(\mathcal{E}_{y_l})}{|\mathcal{E}| - 1}, \exists y_w \in (\mathcal{E} \setminus \mathcal{E}_{y_l}), |\mathcal{E}| \gg |\mathcal{E}_{y_l}| \quad (7)$$

This way, preferred scores $f_\theta(x, y_w)$ and dis-preferred scores $f_\theta(x, y_l)$ are well separated by bout-boundary sampling, which promotes preference optimization. Finally, we demonstrate that out-boundary sampling does not alter the correspondence between the optimization objectives of the unlearning task and the preference optimization task. We propose the following theorem:

Theorem 2 (Preservation of Optimization Objective) Let E_u be the expectation of the unlearning objective with respect to \mathcal{D}_f , and E'_p be the modified preference objective under out-boundary sampling with respect to \mathcal{D}_f^{po} , then we have:

$$E'_p = c'_1 \cdot E_u - c'_2 \quad (8)$$

where $c'_1 > 1$ and $c'_2 > 0$ are constants dependent on the candidate entity set size of \mathcal{D}_f .

See Appendix C for a detailed proof. From Theorem 2, we show that minimizing E'_p remains approximately equivalent to minimizing the original unlearning objective E_u .

3.4 Boundary-Aware Knowledge Recall

In order to preserve remaining knowledge, GraphDPO introduces a boundary-aware knowledge recall mechanism to ensure minimal destruction of remaining knowledge. Specifically, since unlearning mainly affects the forgetting boundary [36], we replay and distill the forgetting boundary.

Boundary Replay. Specifically, we denote \mathcal{D}_{replay} as the set of all triples from \mathcal{B}_{y_l} for each dis-preferred entity y_l in \mathcal{D}_f^{po} . The replay loss on \mathcal{D}_{replay} is defined as follows:

$$\mathcal{L}_{replay} = \mathbb{E}_{(h,r,t) \sim \mathcal{D}_{replay}} [\max(0, f(h, r, t) - f(\bar{h}, r, \bar{t}) + \gamma)] \quad (9)$$

where (\bar{h}, r, \bar{t}) is the negative triple of (h, r, t) with random replacement of head and tail entities, and γ is the margin.

Boundary Distillation. Inspired by recent work on knowledge distillation for KGEs [37, 38], we introduce a distillation strategy to reinforce knowledge retention around the forgetting boundary. Specifically, we define the set of distillation entities $\mathcal{E}_{distill}$ as all boundary neighbors of dis-preferred entities y_l in \mathcal{D}_f^{po} , excluding y_l itself. Following [38], we design the distillation loss for each entity $e \in \mathcal{E}_{distill}$ as follows:

$$\mathcal{L}_e = \begin{cases} \frac{1}{2}(\mathbf{e} - \mathbf{e}_{ref})^2, & |\mathbf{e} - \mathbf{e}_{ref}| \leq 1 \\ |\mathbf{e} - \mathbf{e}_{ref}| - \frac{1}{2}, & |\mathbf{e} - \mathbf{e}_{ref}| > 1 \end{cases} \quad (10)$$

where \mathbf{e} and \mathbf{e}_{ref} denote the embeddings for e in the trained model \mathcal{M} and the reference model \mathcal{M}_{ref} , respectively. Finally, we denote the distillation loss for $\mathcal{E}_{distill}$ as follows:

$$\mathcal{L}_{distill} = \mathbb{E}_{e \sim \mathcal{E}_{distill}} [\mathcal{L}_e] \quad (11)$$

The final optimization object is to minimize \mathcal{L} as follows:

$$\mathcal{L} = \lambda_1 \cdot \mathcal{L}_{dpo} + \lambda_2 \cdot \mathcal{L}_{replay} + \lambda_3 \cdot \mathcal{L}_{distill} \quad (12)$$

where λ_1 , λ_2 , and λ_3 are the weights to balance losses.

4 Experiments

4.1 Experimental Setup

Datasets. We construct eight datasets with 10% and 20% unlearning rate: **FB-10%**, **FB-20%**, **WN-10%**, **WN-20%**, **CO-10%**, **CO-20%**, **YA-10%**, and **YA-20%**, which are based on four datasets with different scales: **FB15K-237** [18], **WN18RR** [19], **CoDEx-L** [20], and **Yago3-10** [21]. Each dataset has 4 unlearning time steps. At each time step i , we construct the forgetting dataset \mathcal{D}_f^i and the remaining dataset \mathcal{D}_r^i . Details of data construction and specific statistics are shown in Appendix D.

Backbones and Baselines. We take **TransE** [27] as the base KGE model in experiments. In exploratory experiments, we also explore other 4 different base KGE models (**TansD** [39], **ComplEx** [28], **SimplEx** [40], **RotatE** [29]) to verify the scalability of our method. We compare our methods with 10 baselines, including: (i) Two exact unlearning methods: **Re-Train** and **Fine-Tune**, which re-train and fine-tune pre-trained KGE models with the whole remaining data \mathcal{D}_r^i at each time step i , respectively. (ii) Two traditional approximate unlearning methods: **RL** [14], **Fisher** [14], and six newest approximate unlearning methods: **BS** [41], **NG** [42], **SSD** [23], **ADV-IMP** [24], **Schema** [25], and **MetaEU** [26]. As one of the approximate unlearning method, our **GraphDPO** mainly compares with approximate methods. Details of the implementations list in Appendix E.

Table 1: Main experimental results on FB-10%, FB-20%, WN-10%, and WN-20%. The bold scores indicate the best results of approximate unlearning methods and underlined scores indicate the second best results in M_{Avg} and M_{F1} . All results are the average of 5 runs and are presented in % form.

	Time 1				Time 2				Time 3				Time 4			
\mathcal{D}_f : FB-10%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
Re-Train	0.203	0.257	0.527	0.388	0.204	0.267	0.532	0.400	0.201	0.275	0.537	0.409	0.201	0.287	0.543	0.422
Fine-Tune	0.239	0.277	0.519	0.406	0.242	0.277	0.518	0.406	0.236	0.282	0.523	0.411	0.224	0.285	0.531	0.417
RL	0.072	0.082	0.505	0.151	0.077	0.082	0.503	0.151	0.092	0.095	0.502	0.172	0.077	0.081	0.502	0.149
Fisher	0.012	0.012	0.500	0.023	0.009	0.009	0.500	0.018	0.007	0.007	0.500	0.014	0.007	0.007	0.500	0.014
BS	0.066	0.075	0.505	0.139	0.067	0.072	0.503	0.134	0.073	0.077	0.502	0.142	0.076	0.080	0.502	0.147
NG	0.143	0.176	0.517	0.292	0.089	0.117	<u>0.512</u>	0.207	0.053	0.074	<u>0.511</u>	0.138	0.035	0.052	<u>0.509</u>	0.099
SSD	0.140	0.160	0.509	0.269	0.140	0.156	0.508	0.265	0.135	0.149	0.507	0.254	0.132	0.146	0.507	0.250
ADV-IMP	0.256	0.257	0.500	0.382	0.227	0.226	0.500	0.350	0.199	0.199	0.499	<u>0.318</u>	0.190	0.189	0.499	<u>0.307</u>
Schema	0.174	0.176	0.501	0.290	0.177	0.174	0.499	0.287	0.184	0.187	0.502	0.304	0.174	0.179	0.503	0.294
MetaEU	0.167	0.163	0.498	0.273	0.164	0.165	0.501	0.276	0.175	0.177	0.501	0.291	0.167	0.166	0.500	0.277
GraphDPO	0.158	0.180	<u>0.511</u>	<u>0.296</u>	0.156	0.184	0.514	<u>0.302</u>	0.154	0.196	0.521	0.319	0.152	0.207	0.527	0.333
\mathcal{D}_f : FB-20%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
Re-Train	0.201	0.281	0.540	0.416	0.210	0.325	0.558	0.461	0.203	0.388	0.593	0.522	0.173	0.541	0.684	0.654
Fine-Tune	0.224	0.277	0.527	0.408	0.206	0.276	0.535	0.410	0.197	0.301	0.552	0.438	0.184	0.376	0.596	0.515
RL	0.074	0.083	0.504	0.153	0.076	0.081	0.503	0.149	0.085	0.088	0.501	0.160	0.075	0.078	0.502	0.144
Fisher	0.012	0.012	0.500	0.024	0.009	0.010	0.500	0.012	0.009	0.009	0.500	0.017	0.008	0.008	0.500	0.016
BS	0.069	0.078	0.505	0.145	0.071	0.076	0.503	0.140	0.067	0.071	0.502	0.131	0.069	0.072	0.502	0.134
NG	0.086	0.106	0.510	0.189	0.036	0.053	0.509	0.100	0.019	0.029	0.505	0.057	0.014	0.020	0.503	0.039
SSD	0.157	0.178	<u>0.511</u>	0.294	0.152	0.172	<u>0.510</u>	0.285	0.143	0.164	<u>0.511</u>	0.276	0.132	0.163	<u>0.516</u>	0.274
ADV-IMP	0.287	0.195	0.454	<u>0.306</u>	0.262	0.191	0.464	<u>0.303</u>	0.224	0.182	0.479	<u>0.295</u>	0.197	0.175	0.489	0.287
Schema	0.168	0.172	0.502	0.285	0.165	0.176	0.506	0.291	0.162	0.178	0.508	0.294	0.158	0.181	0.512	0.298
MetaEU	0.162	0.167	0.503	0.278	0.160	0.174	0.507	0.288	0.157	0.179	<u>0.511</u>	<u>0.295</u>	0.154	0.183	0.515	<u>0.301</u>
GraphDPO	0.156	0.189	0.517	0.309	0.154	0.191	0.518	<u>0.311</u>	0.151	0.218	0.521	<u>0.347</u>	0.150	0.227	0.527	<u>0.333</u>
\mathcal{D}_f : WN-10%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
Re-Train	0.182	0.596	0.707	0.689	0.165	0.609	0.722	0.704	0.146	0.630	0.742	0.725	0.167	0.648	0.761	0.744
Fine-Tune	0.222	0.554	0.666	0.647	0.252	0.501	0.625	0.600	0.162	0.422	0.630	0.562	0.171	0.429	0.629	0.566
RL	0.100	0.118	<u>0.509</u>	0.208	0.064	0.066	0.501	0.123	0.044	0.042	0.499	0.080	0.020	0.019	0.499	0.037
Fisher	0.022	0.014	0.496	0.028	0.022	0.010	0.493	0.017	0.021	0.006	0.492	0.013	0.020	0.004	0.492	0.008
BS	0.093	0.105	0.506	0.188	0.017	0.013	0.498	0.026	0.009	0.008	0.499	0.016	0.007	0.007	0.500	0.013
NG	0.510	0.361	0.426	<u>0.416</u>	0.507	0.362	0.427	<u>0.417</u>	0.499	0.361	0.431	<u>0.419</u>	0.501	0.361	0.430	<u>0.419</u>
SSD	0.156	0.190	0.513	0.309	0.063	0.150	0.544	0.259	0.061	0.116	<u>0.527</u>	0.207	0.047	0.116	<u>0.534</u>	0.206
ADV-IMP	0.176	0.173	0.498	0.286	0.059	0.058	0.500	0.109	0.052	0.051	0.499	0.097	0.046	0.047	0.502	0.088
Schema	0.286	0.224	0.469	0.341	0.292	0.217	0.465	0.332	0.284	0.222	0.469	0.339	0.312	0.242	0.465	0.358
MetaEU	0.292	0.211	0.460	0.325	0.302	0.204	0.451	0.316	0.297	0.211	0.457	0.325	0.324	0.254	0.465	0.369
GraphDPO	0.367	0.323	0.479	0.428	0.304	0.320	<u>0.508</u>	0.438	0.283	0.340	0.528	<u>0.461</u>	0.376	0.272	0.552	<u>0.496</u>
\mathcal{D}_f : WN-20%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
Re-Train	0.166	0.616	0.725	0.709	0.127	0.644	0.759	0.741	0.080	0.679	0.800	0.781	0.026	0.762	0.868	0.855
Fine-Tune	0.210	0.572	0.681	0.664	0.238	0.513	0.638	0.613	0.131	0.479	0.674	0.618	0.172	0.526	0.677	0.643
RL	0.135	0.140	0.501	0.035	0.020	0.019	0.499	0.038	0.010	0.011	0.500	0.021	0.013	0.014	0.500	0.027
Fisher	0.024	0.022	0.499	0.042	0.006	0.005	0.499	0.009	0.004	0.005	0.500	0.010	0.004	0.002	0.499	0.004
BS	0.039	0.037	0.499	0.072	0.009	0.009	0.500	0.017	0.006	0.007	0.500	0.014	0.006	0.007	0.500	0.014
NG	0.513	0.407	0.447	<u>0.443</u>	0.513	0.403	0.441	<u>0.445</u>	0.515	0.403	0.444	<u>0.440</u>	0.517	0.405	0.494	<u>0.441</u>
SSD	0.058	0.160	0.519	0.085	0.057	0.124	<u>0.533</u>	0.218	0.042	0.137	0.547	0.240	0.030	0.144	0.557	0.250
ADV-IMP	0.157	0.158	0.500	0.266	0.143	0.145	0.501	0.248	0.137	0.139	0.501	0.239	0.095	0.099	0.502	0.178
Schema	0.172	0.208	0.518	0.332	0.184	0.223	0.520	0.350	0.128	0.234	<u>0.553</u>	0.369	0.101	0.294	<u>0.597</u>	0.443
MetaEU	0.184	0.224	<u>0.520</u>	0.352	0.193	0.234	0.521	0.363	0.134	0.219	<u>0.543</u>	0.350	0.087	0.259	0.586	0.404
GraphDPO	0.220	0.319	0.549	0.452	0.258	0.325	0.534	0.451	0.143	0.352	0.605	0.499	0.118	0.433	0.658	0.581

Metrics. We evaluate the performance of the model on the link prediction task and use the Mean Reciprocal Rank (MRR) as the evaluation metric. At each time step i , we test the model on both the accumulated forgetting dataset $\sum_{j=1}^i \mathcal{D}_f^j$ and the remaining dataset \mathcal{D}_r^i . To evaluate the unlearning performance and preserving performance together, we denote $MRR_{Avg}^i = \frac{MRR_r^i + (1 - MRR_f^i)}{2}$ and $MRR_{F1}^i = \frac{2 \cdot MRR_r^i \cdot (1 - MRR_f^i)}{MRR_r^i + (1 - MRR_f^i)}$, where MRR_r^i and MRR_f^i denote the MRR metric on \mathcal{D}_r^i and $\sum_{j=1}^i \mathcal{D}_f^j$, respectively. Higher MRR_{Avg}^i and MRR_{F1}^i indicate better performance.

4.2 Results

Main Results. Main results on FB-10%, FB-20%, WN-10%, and WN-20% are presented in Table 1, and the results on CO-10%, CO-20%, YA-10%, YA-20% are shown in Appendix G. First, compared to the exact unlearning methods, Re-Train and Fine-Tune, GraphDPO achieves 93%-99%, 82%-97%, 68%-88%, and 69%-97% of their performance in MRR_{Avg} across all datasets. Similarly, GraphDPO achieves 73%-80%, 63%-81%, 62%-88%, and 61%-90% of their performance in MRR_{F1} . It shows that GraphDPO with only partial data closely approximates the performance of full-data training.

Second, compared to approximate unlearning baselines, GraphDPO achieves the best MRR_{Avg} and MRR_{F1} on most datasets. On FB-20% and WN-20%, GraphDPO outperforms other methods by 0.1%-10.1% in MRR_{Avg} and 0.1%-14.0% in MRR_{F1} , demonstrating its strength in forgetting large-scale knowledge while preserving core information. On FB-10% and WN-10%, where the forgetting rate is lower, GraphDPO remains competitive: slightly underperforming in Time 1 and 2

Table 2: Ablation experimental results on FB-20% and WN-20%. Replay, Dis, and O-S denote boundary replay, boundary distillation and out-boundary sampling, respectively. All results are the average of 5 runs.

	Time 1				Time 2				Time 3				Time 4			
\mathcal{D}_f : FB-20%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
GraphDPO	0.156	0.189	0.517	0.309	0.154	0.191	0.518	0.311	0.151	0.218	0.534	0.347	0.150	0.275	0.562	0.415
w/o DPO	0.130	0.141	<u>0.505</u>	0.242	0.153	0.176	0.511	0.230	0.164	0.204	0.520	0.327	0.167	0.245	0.539	0.378
w/o Replay	0.139	0.097	0.479	0.175	0.086	0.079	0.496	0.145	0.076	0.074	0.499	0.138	0.074	0.074	0.500	0.138
w/o Dis	0.131	0.141	<u>0.505</u>	0.242	0.118	0.154	<u>0.517</u>	0.262	0.117	0.179	<u>0.529</u>	0.297	0.119	0.221	<u>0.551</u>	0.353
w/o O-S	0.210	0.179	0.485	<u>0.292</u>	0.168	0.160	0.496	<u>0.268</u>	0.149	0.154	0.503	0.261	0.138	0.158	0.510	0.266
\mathcal{D}_f : WN-20%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
GraphDPO	0.220	0.319	0.549	0.452	0.258	0.325	0.534	0.451	0.143	0.352	0.605	0.499	0.118	0.433	0.658	0.581
w/o DPO	0.276	0.245	0.485	0.366	0.282	0.302	0.510	0.425	0.284	0.384	0.549	0.498	0.279	0.485	0.603	0.579
w/o Replay	0.443	0.273	0.415	0.366	0.303	0.156	0.427	0.255	0.079	0.038	0.479	0.072	0.067	0.032	0.482	0.061
w/o Dis	0.360	0.278	0.459	0.388	0.286	0.308	<u>0.511</u>	0.431	0.093	0.284	<u>0.596</u>	0.433	0.121	0.394	<u>0.637</u>	0.544
w/o O-S	0.440	0.306	0.433	<u>0.395</u>	0.359	0.338	0.490	<u>0.443</u>	0.338	0.401	0.531	0.498	0.207	0.396	0.595	0.528

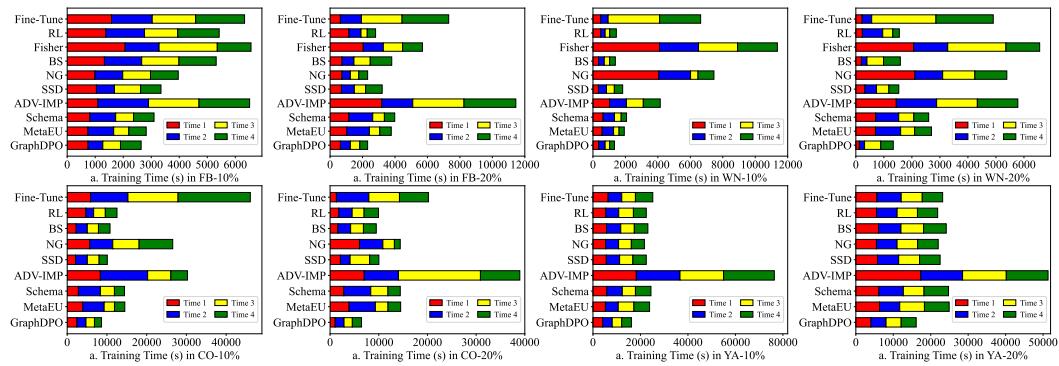


Figure 3: Time efficiency analysis. Fisher fails to run in the last four datasets with out-of-memory.

(-1.8% MRR_{Avg} , -2.5% MRR_{F1} on average), but outperforming in Time 3 and 4 (+1.2% MRR_{Avg} , +3.7% MRR_{F1}). This indicates that while low forgetting rates limit immediate gains, GraphDPO maintains strong performance in continual unlearning.

Third, we observe an unbalance between forgetting and retention in some baselines: Fisher achieves strong forgetting (e.g., 1.2% drop in M_f on FB-20%) but also incur similar retention loss (1.2% drop in M_r). Conversely, methods like NG retain more knowledge but forget less, for example, M_f remains high at 51.0% and 51.3% on WN-10% and WN-20%, respectively. In contrast, GraphDPO achieves a better balance, with average MRR_{Avg} and MRR_{F1} reaching 53.5% and 41.1%, respectively.

Fourth, we observe that on large-scale KGs (CO-10%, CO-20%, YA-10%, YA-20%), our GraphDPO consistently outperforms all baselines in both MRR_{Avg} and MRR_{F1} . It demonstrates not only the scalability of our approach to large KGs, but also its superior performance compared to existing methods. In particular, when compared to two KGE-specific unlearning methods, Schema and MetaEU, GraphDPO achieves gains of 1.0%–6.2% in MRR_{Avg} and 2.3%–8.2% in MRR_{F1} , respectively.

Ablation Results. To assess the effectiveness of each module in GraphDPO, we conduct ablation experiments on its full version and variants. Results on FB-20% and WN-20% are shown in Table 2, with additional results in Appendix H. Removing any module leads to a performance drop of 0.1%–17.6% in M_{Avg} and 0.2%–52.0% in M_{F1} , confirming their collective importance.

In particular, removing the Replay module causes a 4.6%–40.1% drop in M_r , highlighting its key role in preserving retained knowledge. Removing the O-S module results in a 1.4% and 12.6% increase in M_f on FB-20% and WN-20%, respectively, indicating its effectiveness in improving forgetting.

Time Efficiency. To evaluate the time efficiency, we compare the training time of all methods as shown in Figure 3. First, compared to the exact unlearning method Fine-Tune, GraphDPO saves 69%–80% of the training time on all datasets, highlighting its significant advantage over exact unlearning.

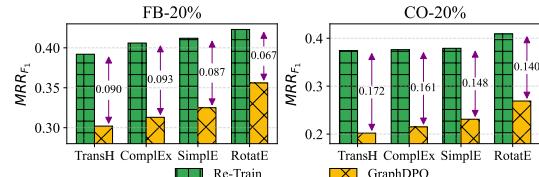


Figure 4: Scalability of different KGE models.

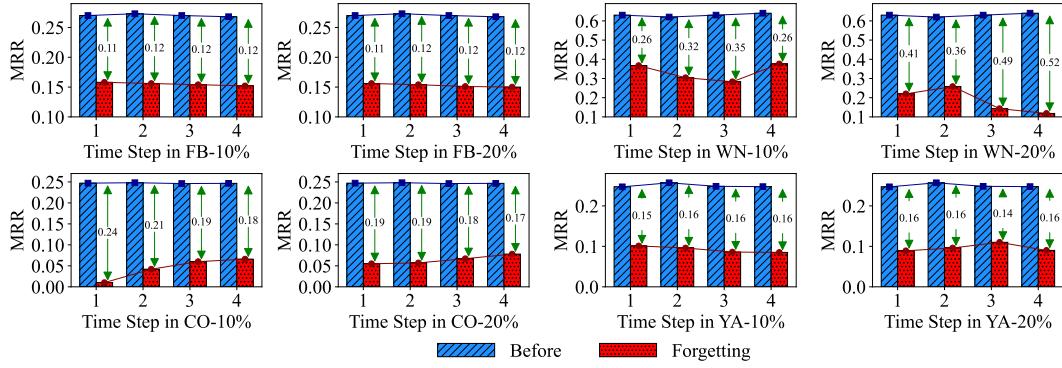


Figure 5: Forgetting analyse of GraphDPO. *Before* defines the performance of original pre-trained model on \mathcal{D}_f , and *Forgetting* defines the performance of GraphDPO on \mathcal{D}_f .

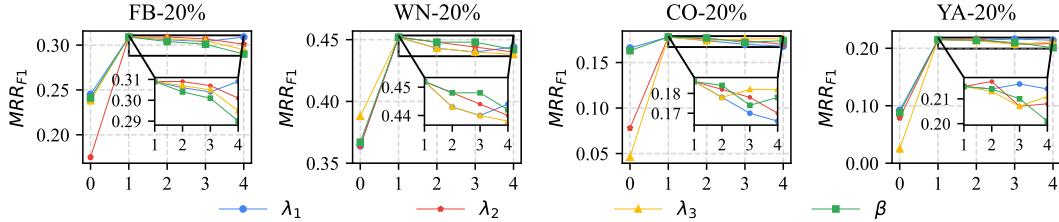


Figure 6: Hyperparametric analysis of λ_1 , λ_2 , λ_3 , and β with value from 0 to 4 in one time step.

Second, GraphDPO achieves the fastest training speed compared to approximate unlearning methods, reducing training time by 1%-88%. It proves that GraphDPO is more efficient than other baselines.

Scalability of Different KGE Models. To assess the scalability of our method across different KGE models, we implement GraphDPO in TransH, ComplEx, SimplE, and RotatE on FB-20% and CO-20%, and compare it with full retraining in Figure 4. GraphDPO achieves 77%-84% of the full retraining performance on FB-20% and 54%-66% on CO-20%, demonstrating its effectiveness across various KGE backbones. Moreover, we notice that the performance gap between GraphDPO and retraining as the underlying KGE model improves (e.g., from 9.0% to 6.07% on FB-20%, and 17.2% to 14.0% on CO-20%), suggesting that GraphDPO benefits more from stronger base models.

Forgetting Analyse. To investigate the forgetting capability of GraphDPO, we compare the effectiveness of GraphDPO on forgetting datasets with original pre-trained models, as shown in Figure 5. First, compared to original models, performance decreases notably by 10%-50% on all datasets in MRR_f after optimization by GraphDPO, demonstrating its significant forgetting performance. Second, the performance of GraphDPO on forgetting datasets decreases as time goes on, dropping from 0.16% to 0.14% and from 0.32% to 0.13%, reflecting its ability for continual forgetting.

Hyperparametric Analysis. To assess hyperparameter sensitivity, we select one time step from each of four datasets (FB-20%, WN-20%, CO-20%, YA-20%) and tune λ_1 , λ_2 , λ_3 , and β . Results are shown in Figure 6. First, we find that the performance drops notably when any hyperparameter is set to 0 and peaks at 1, indicating their necessity. Beyond 1, performance changes are minor (1%-2%), suggesting that our GraphDPO is robust and works well with default values set to 1.

Visualization Analyse. To verify that our method can solve the issues of incomplete knowledge removal and the weakening of remaining knowledge, we use the t-SNE [43] to visualize the embeddings of both forgetting entities and those on the preserved boundary, as shown in Figure 7. (1) For forgetting entities (marked by asterisks), GraphDPO makes a substantial shift in their embeddings compared to Finetune. This indicates that GraphDPO effectively addresses the incomplete knowledge removal problem. (2) For entities on the forgetting

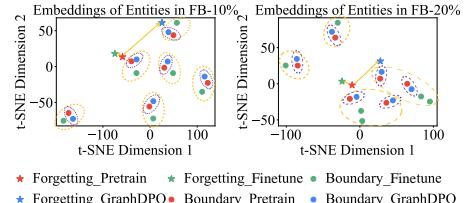


Figure 7: Visualization of forgetting and boundary entity embeddings.

boundary (marked by circles), GraphDPO induces minimal changes in the embeddings relative to Finetune. It shows that GraphDPO can mitigate the destruction of the remaining knowledge.

5 Conclusion

In this paper, we dive into the task of unlearning for KGE. To address the challenge of forgetting knowledge in KGs due to their connectivity, we propose GraphDPO, which utilizes a graph-aware direct preference optimization algorithm with out-boundary sampling for effective forgetting learning. We also introduce boundary-aware knowledge recall to ensure that the remaining knowledge is preserved. In the future, we will explore more efficient unlearning methods on graphical models. More detailed limitations and future directions of unlearning in KGE are provided in Appendix F.

References

- [1] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.
- [2] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–49, 2021.
- [3] Xin Dong, Evgeniy Gabrilovich, Jeremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*, 2014.
- [4] Antoine Bordes, Jason Weston, and Nicolas Usunier. Open question answering with weakly supervised embedding models. In *ECML-PKDD*, 2014.
- [5] Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In *ACL*, 2014.
- [6] Joel Barmettler, Abraham Bernstein, and Luca Rossetto. Conceptformer: Towards efficient use of knowledge-graph embeddings in large language models. *arXiv preprint arXiv:2504.07624*, 2025.
- [7] Anwar Said, Tyler Derr, Mudassir Shabbir, Waseem Abbas, and Xenofon Koutsoukos. A survey of graph unlearning. *arXiv preprint arXiv:2310.02164*, 2023.
- [8] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *WWW*, 2007.
- [9] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 2013.
- [10] Siyuan Cheng, Ningyu Zhang, Bozhong Tian, Xi Chen, Qingbin Liu, and Huajun Chen. Editing language model-based knowledge graph embeddings. In *AAAI*, 2024.
- [11] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2021.
- [12] Ronen Eldan and Mark Russinovich. Who’s harry potter? approximate unlearning in llms. *arXiv preprint arXiv:2310.02238*, 2023.
- [13] Weiqi Wang, Zhiyi Tian, and Shui Yu. Machine unlearning: A comprehensive survey. *arXiv preprint arXiv:2405.07406*, 2024.
- [14] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *CVPR*, 2020.
- [15] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 303–319. IEEE, 2022.

- [16] Haonan Yan, Xiaoguang Li, Ziyao Guo, Hui Li, Fenghua Li, and Xiaodong Lin. Arcane: An efficient architecture for exact machine unlearning. In *IJCAI*, 2022.
- [17] Ga Wu, Masoud Hashemi, and Christopher Srinivasa. Puma: Performance unchanged model augmentation for training data removal. In *AAAI*, 2022.
- [18] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.
- [19] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, 2015.
- [20] Tara Safavi and Danai Koutra. Codex: A comprehensive knowledge graph completion benchmark. In *EMNLP*, pages 8328–8350, 2020.
- [21] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*, 2013.
- [22] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*, 2022.
- [23] Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *AAAI*, 2024.
- [24] Sungmin Cha, Sungjun Cho, Dasol Hwang, Honglak Lee, Taesup Moon, and Moontae Lee. Learning to unlearn: Instance-wise unlearning for pre-trained classifiers. In *AAAI*, 2024.
- [25] Yang Xiao, Ruimeng Ye, and Bo Hui. Knowledge graph unlearning with schema. In *COLING*, pages 3541–3546, 2025.
- [26] Naixing Xu, Qian Li, Xu Wang, Bingchen Liu, and Xin Li. Learn to unlearn: Meta-learning-based knowledge graph embedding unlearning. *arXiv preprint arXiv:2412.00881*, 2024.
- [27] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.
- [28] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, 2016.
- [29] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.
- [30] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- [31] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*, 2024.
- [32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [33] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2024.
- [34] Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *AISTATS*, 2024.
- [35] Ziyu Shang, Peng Wang, Wenjun Ke, Jiajun Liu, Hailang Huang, Guozheng Li, Chenxiao Wu, Jianghan Liu, Xiye Chen, and Yining Li. Learning multi-granularity and adaptive representation for knowledge graph reasoning. In *IJCAI*, 2024.

- [36] Zhenyi Wang, Enneng Yang, Li Shen, and Heng Huang. A comprehensive survey of forgetting in deep learning beyond continual learning, 2023.
- [37] Yushan Zhu, Wen Zhang, Mingyang Chen, Hui Chen, Xu Cheng, Wei Zhang, and Huajun Chen. Dualde: Dually distilling knowledge graph embedding for faster and cheaper reasoning. In *WSDM*, 2022.
- [38] Jiajun Liu, Wenjun Ke, Peng Wang, Ziyu Shang, Jinhua Gao, Guozheng Li, Ke Ji, and Yanhe Liu. Towards continual knowledge graph embedding via incremental distillation. In *AAAI*, 2024.
- [39] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.
- [40] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. *NeurIPS*, 2018.
- [41] Min Chen, Weizhuo Gao, Gaoyang Liu, Kai Peng, and Chen Wang. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In *CVPR*, 2023.
- [42] Yuanshun Yao, Xiaojun Xu, and Yang Liu. Large language model unlearning. *arXiv preprint arXiv:2310.10683*, 2023.
- [43] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- [44] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.
- [45] Kai Wang, Yu Liu, Qian Ma, and Quan Z Sheng. Mulde: Multi-teacher knowledge distillation for low-dimensional knowledge graph embeddings. In *WWW*, pages 1716–1726, 2021.
- [46] Jiajun Liu, Peng Wang, Ziyu Shang, and Chenxiao Wu. Iterde: an iterative knowledge distillation framework for knowledge graph embeddings. In *AAAI*, 2023.
- [47] Mingyang Chen, Wen Zhang, Zhen Yao, Yushan Zhu, Yang Gao, Jeff Z Pan, and Huajun Chen. Entity-agnostic representation learning for parameter-efficient knowledge graph embedding. In *AAAI*, 2023.
- [48] Angel Daruna, Mehul Gupta, Mohan Sridharan, and Sonia Chernova. Continual learning of knowledge graph embeddings. *IEEE Robotics and Automation Letters*, 6(2):1128–1135, 2021.
- [49] Yuanning Cui, Yuxin Wang, Zequn Sun, Wenqiang Liu, Yiqiao Jiang, Kexin Han, and Wei Hu. Lifelong embedding learning and transfer for growing knowledge graphs. In *AAAI*, 2023.
- [50] Yash Sinha, Murari Mandal, and Mohan Kankanhalli. Distill to delete: unlearning in graph networks with knowledge distillation. *arXiv preprint arXiv:2309.16173*, 2023.
- [51] Chao Pan, Eli Chien, and Olgica Milenkovic. Unlearning graph classifiers with limited data resources. In *WWW*, pages 716–726, 2023.
- [52] Jiahao Zhang. Graph unlearning with efficient partial retraining. In *Companion Proceedings of the ACM Web Conference 2024*, pages 1218–1221, 2024.
- [53] Yash Sinha, Murari Mandal, and Mohan Kankanhalli. Distill to delete: Unlearning in graph networks with knowledge distillation, 2024.
- [54] Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*, 2024.
- [55] Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. In *ICML*, pages 58348–58365, 2024.

- [56] Yuzhe Gu, Wenwei Zhang, Chengqi Lyu, Dahua Lin, and Kai Chen. Mask-dpo: Generalizable fine-grained factuality alignment of llms. *arXiv preprint arXiv:2503.02846*, 2025.
- [57] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, and et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

A Detailed Related Work

This section outlines the most relevant research of three key areas in detail: knowledge graph embedding, knowledge unlearning, and preference optimization.

Knowledge Graph Embedding. Knowledge graph embedding (KGE) aims to represent entities and relations in a continuous vector space while preserving the structural information of the graph. Early translational models such as TransE [27] interpret a relation as a translation vector between head and tail entities. Extensions like TransH [39] and TransR [44] introduce relation-specific hyperplanes or spaces to better model complex relations. Later, RotatE [29] represents relations as rotations in the complex vector space, capturing various relational patterns including symmetry and inversion. Recent studies have explored KGE methods under resource-constrained settings [45, 37, 46, 47]. In parallel, continual and incremental knowledge graph embedding has gained attention, aiming to support dynamically evolving knowledge without retraining from scratch [48, 49, 38]. However, much less attention has been paid to the task of knowledge unlearning [26, 25], which requires selectively forgetting specific facts or entities in the graph while preserving the rest of the knowledge. This work aims to bridge this gap by explicitly modeling unlearning in KGE.

Knowledge Unlearning. Knowledge unlearning aims to remove incorrect and outdated knowledge from machine learning models [13]. Existing unlearning methods can be categorized into exact and approximate approaches. Exact unlearning methods [14, 22] require re-training the model using all reserved data, resulting in significant training costs. To mitigate this issue, approximate unlearning methods [15, 17] update models using only the forgetting datasets, with little or no use of the reserved datasets, thereby reducing training time [23, 24]. Recently, several unlearning algorithms have been proposed for applications in graph learning [50, 51, 52, 53]. Recent works utilize schema [25] or meta-learning [26] to forget knowledge in KGE models, however, they struggle to effectively forget while preserving remaining knowledge [27, 28, 29] due to the inherent connectivity of KGs.

Preference Optimization. Preference optimization seeks to align LLMs with human preferences and values [30, 54, 55, 56], and can be categorized into online and offline algorithms [31]. Online algorithms incorporate reinforcement learning with supervised fine-tuning and policy optimization, which are inherently complex and challenging to optimize [32]. To solve these issues, offline algorithms like DPO [33] directly compare different decision sequences to optimize models, resulting in more efficient performance [34]. As alignment techniques leverage both positive and negative samples, offline algorithms are well-suited for selectively forgetting and remaining knowledge. In this paper, we demonstrate the effectiveness of applying DPO for unlearning in KGs.

B Proof of Theorem 1: Equivalence of Optimization Objectives

In this section, we prove the Equivalence of Minimizing E_p and E_u . For the unlearning task based on the forgetting dataset \mathcal{D}_f , the optimization objective is to minimize E_u as:

$$E_u = \mathbb{E}_{(h, r, t) \sim \mathcal{D}_f} [f(h, r, t)], \quad (13)$$

where h , r , t denote head entity, relation, and tail entity in a forgetting triple $(h, r, t) \in \mathcal{D}_f$, respectively. $f(\cdot)$ is the score function of KGE models.

For the preference optimization task based on \mathcal{D}_f^{po} , the optimization objective is to minimize E_p as:

$$\begin{aligned} E_p &= \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}_f^{po}} [f(x, y_l) - f(x, y_w)] \\ &= \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}_f^{po}} [f(x, y_l)] - \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}_f^{po}} [f(x, y_w)] \end{aligned} \quad (14)$$

where x , y_w , and y_l denote the query, preferred entity, and dispreferred entity in $(x, y_w, y_l) \in \mathcal{D}_f^{po}$, respectively. For the first term in Equation 14, we have:

$$\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}_f^{po}} f(x, y_l) = \mathbb{E}_{(h, r, t) \sim \mathcal{D}_f} [f(h, r, t)] = E_u \quad (15)$$

For the second term in Equation 14, we analyze the expectation over the sampled y_w . Since y_w is uniformly sampled from the candidate set $\mathcal{E} \setminus t$, we have:

$$\mathbb{E}_{(x,y_w,y_l) \sim \mathcal{D}_f^{po}}[f(x,y_w)] = \frac{1}{|\mathcal{E}| - 1} [C - \mathbb{E}_{(x,y_w,y_l) \sim \mathcal{D}_f^{po}} f(x,y_l)], \quad (16)$$

where $C = \mathbb{E}_{(x,y_w,y_l) \in \mathcal{D}_f^{po}} \sum_{y \in \mathcal{E}} [f(x,y)]$. Then, we substitute Equation 15 and Equation 16 into Equation 14, we have:

$$\begin{aligned} E_p &= \mathbb{E}_{(x,y_w,y_l) \sim \mathcal{D}_f^{po}} f(x,y_l) - \frac{C - \mathbb{E}_{(x,y_w,y_l) \sim \mathcal{D}_f^{po}} f(x,y_l)}{|\mathcal{E}| - 1} \\ &= \frac{|\mathcal{E}|}{|\mathcal{E}| - 1} \cdot \mathbb{E}_{(x,y_w,y_l) \sim \mathcal{D}_f^{po}} f(x,y_l) - \frac{C}{|\mathcal{E}| - 1} \\ &= c_1 \cdot E_u - c_2 \end{aligned} \quad (17)$$

where $c_1 = \frac{|\mathcal{E}|}{|\mathcal{E}| - 1} > 1$ is the positive constant, and $c_2 = \frac{C}{|\mathcal{E}| - 1} > 0$. Notice that $0 < f(x,y) < 1$, we have:

$$0 < c_2 = \frac{\mathbb{E}_{(x,y_w,y_l) \in \mathcal{D}_f^{po}} \sum_{y \in \mathcal{E}} [f(x,y)]}{|\mathcal{E}| - 1} < \frac{\mathbb{E}_{(x,y_w,y_l) \in \mathcal{D}_f^{po}} |\mathcal{E}|}{|\mathcal{E}| - 1} = \frac{|\mathcal{E}|}{|\mathcal{E}| - 1} = c_1 \quad (18)$$

Therefore, we have that $0 < c_2 < c_1$. From Equation 17, we observe that minimizing E_u is approximately equivalent to minimizing E_p .

C Proof of Theorem 2: Preservation of Optimization Objective

In this section, we prove the equivalence of minimizing E'_p and E_u . Combined with Equation 7, the modified preference objective E'_p can be calculated as:

$$\begin{aligned} E'_p &= \mathbb{E}_{(x,y_w,y_l) \sim \mathcal{D}_f^{po}} [f(x,y_l)] - \mathbb{E}_{(x,y_w,y_l) \sim \mathcal{D}_f^{po}} [f(x,y_w)] \\ &= E_u - \frac{C - \sum_{e \in \mathcal{E}_{y_l}} \mathbb{E}_{(x,y_w,e) \sim \mathcal{D}_f^{po}} [f(x,e)]}{|\mathcal{E}| - 1} \\ &= \frac{|\mathcal{E}|}{|\mathcal{E}| - 1} \cdot E_u - \frac{C - \sum_{e \in \mathcal{E}_{y_l}} \mathbb{E}_{(x,y_w,e) \sim \mathcal{D}_f^{po}} [f(x,e)]}{|\mathcal{E}| - 1} \\ &= c'_1 \cdot E_u - c'_2 \end{aligned} \quad (19)$$

where $C = \mathbb{E}_{(x,y_w,y_l) \in \mathcal{D}_f^{po}} \sum_{y \in \mathcal{E}} [f(x,y)]$, $c'_1 = \frac{|\mathcal{E}|}{|\mathcal{E}| - 1} > 1$, $c'_2 = \frac{C - \sum_{e \in \mathcal{E}_{y_l}} \mathbb{E}_{(x,y_w,e) \sim \mathcal{D}_f^{po}} [f(x,e)]}{|\mathcal{E}| - 1}$. Since we have already shown in Equation 7 that $|\mathcal{E}| \gg |\mathcal{E}_{y_l}|$, and the values of $f(x,e)$ are bounded, the aggregated expectation over \mathcal{E}_{y_l} becomes negligible compared to the total sum C . Therefore, the correction term in c'_2 is very small, and we approximate:

$$c'_2 \approx \frac{C}{|\mathcal{E}| - 1} \quad (20)$$

Similary to Equation 18, we have:

$$0 < c'_2 = \frac{\mathbb{E}_{(x,y_w,y_l) \in \mathcal{D}_f^{po}} \sum_{y \in \mathcal{E}} [f(x,y)]}{|\mathcal{E}| - 1} < \frac{\mathbb{E}_{(x,y_w,y_l) \in \mathcal{D}_f^{po}} |\mathcal{E}|}{|\mathcal{E}| - 1} = \frac{|\mathcal{E}|}{|\mathcal{E}| - 1} = c'_1 \quad (21)$$

Therefore, we have that $0 < c'_2 < c'_1$. Thus, $E'_p = c'_1 \cdot E_u - c'_2$ still holds, and the equivalence is preserved.

D Dataset Construction and Statistics

D.1 Dataset Construction

We construct eight datasets based on four datasets **FB15K-237** [18], **WN18RR** [19], **CoDEx-L** [20], and **Yago3-10** [21], which contains different scales of KGs, with 10% and 20% unlearning rate: **FB-10%**, **FB-20%**, **WN-10%**, **WN-20%**, **CO-10%**, **CO-20%**, **YA-10%**, **YA-20%**, which contain 10% or 20% triples that need to be forgotten. In order to further simulate the continuously evolving KGs in the real world, each dataset is set up with 4 time steps. At each time step i , we construct the forgetting dataset \mathcal{D}_f^i and the remaining dataset \mathcal{D}_r^i . The \mathcal{D}_f^i is drawn from the origin dataset \mathcal{D}_{ori} by uniform sampling with the unlearning rate, and the $\mathcal{D}_r^i = \mathcal{D}_{ori} - \sum_{j=1}^i \mathcal{D}_f^j$.

During dataset construction, at each time step, we sample both connected and unconnected triples with respect to the existing oblivion set. This ensures that each constructed oblivion set includes both triples that are semantically entangled with the retained knowledge, as well as those that are structurally independent.

The complete sampling procedure is illustrated in Algorithm 1, which incrementally builds the oblivion and remaining sets by iteratively identifying candidate triples, separating them by connectivity, and selecting a balanced subset at each step.

Algorithm 1: Controlled Unlearning Dataset Construction

```

Input: Original triple set  $\mathcal{D}_{ori}$ , total unlearning rate  $r$ , number of time steps  $T$ 
Output: Forgetting datasets  $\{\mathcal{D}_f^1, \dots, \mathcal{D}_f^T\}$  and remaining datasets  $\{\mathcal{D}_r^1, \dots, \mathcal{D}_r^T\}$ 
Initialize historical forgetting set:  $\mathcal{D}_{hist} \leftarrow \emptyset$ ;
for  $t = 1$  to  $T$  do
     $\mathcal{C} \leftarrow \mathcal{D}_{ori} \setminus \mathcal{D}_{hist}$ ;                                // Remaining candidates
     $\mathcal{E}_{hist} \leftarrow$  all entities in  $\mathcal{D}_{hist}$ ;
     $\mathcal{C}_{conn} \leftarrow \{(h, r, t) \in \mathcal{C} \mid h \in \mathcal{E}_{hist} \vee t \in \mathcal{E}_{hist}\}$ ;
     $\mathcal{C}_{unconn} \leftarrow \mathcal{C} \setminus \mathcal{C}_{conn}$ ;
    Sample  $r \cdot |\mathcal{C}_{conn}|$  triples from  $\mathcal{C}_{conn}$  as  $\mathcal{D}_{f,conn}^t$ ;
    Sample  $r \cdot |\mathcal{C}_{unconn}|$  triples from  $\mathcal{C}_{unconn}$  as  $\mathcal{D}_{f,unconn}^t$ ;
     $\mathcal{D}_f^t \leftarrow \mathcal{D}_{f,conn}^t \cup \mathcal{D}_{f,unconn}^t$ ;
     $\mathcal{D}_r^t \leftarrow \mathcal{D}_{ori} \setminus \bigcup_{i=1}^t \mathcal{D}_f^i$ ;
     $\mathcal{D}_{hist} \leftarrow \mathcal{D}_{hist} \cup \mathcal{D}_f^t$ ;

```

Compared to the datasets constructed in previous work [25, 26] for the unlearning of KGE, our datasets have the following advantages, which simulate different real-world situations: (1) Covering various scales of KGs (from 92,583 triples to 1,089,000 triples). (2) Covering various connections between forgetting triples and remaining triples (forgetting triples of connecting and disconnecting to remaining triples both exist). (3) Covering various time steps, which simulates continual unlearning.

D.2 Dataset Statistics

The statistics of the origin datasets **FB15k-237**, **WN18RR**, **CoDEx-L**, and **YAGO3-10** are shown in Table 3. The statistics of the constructed datasets **FB-10%**, **FB-20%**, **WN-10%**, **WN-20%**, **CO-10%**, **CO-20%**, **YA-10%**, and **YA-20%** are shown in Table 4.

Table 3: The statistics of the origin datasets.

Dataset	Entity Num	Relation Num	Triple Num
FB15k-237	14,505	237	310,079
WN18RR	40,559	11	92,583
CoDEx-L	77,951	69	612,437
YAGO3-10	123,143	37	1,089,000

Table 4: The statistics of the constructed datasets. $|\mathcal{E}|$, $|\mathcal{R}|$ and $|\mathcal{T}|$ denote the number of entities, relations and triples in the origin KG \mathcal{D}_{ori} . $|\mathcal{D}_f^i|$ and $|\mathcal{D}_r^i|$ denote the number of forgetting triples and remaining triples in i -th time step, respectively.

Dataset	\mathcal{D}_{ori}			Time 1		Time 2		Time 3		Time 4	
	$ \mathcal{E} $	$ \mathcal{R} $	$ \mathcal{T} $	$ \mathcal{D}_f^1 $	$ \mathcal{D}_r^1 $	$ \mathcal{D}_f^2 $	$ \mathcal{D}_r^2 $	$ \mathcal{D}_f^3 $	$ \mathcal{D}_r^3 $	$ \mathcal{D}_f^4 $	$ \mathcal{D}_r^4 $
FB-10%	14,541	237	310,116	31,011	279,105	31,011	248,094	31,011	217,083	31,011	186,072
FB-20%	14,541	237	310,116	62,023	248,093	62,023	186,070	62,023	124,047	62,023	62,024
WN-10%	40,943	11	93,003	9,300	83,703	9,300	74,403	9,300	65,103	9,300	55,803
WN-20%	40,943	11	93,003	18,600	74,403	18,600	55,803	18,600	37,203	18,600	18,603
CO-10%	77,951	69	612,437	61,243	551,194	612,43	489,951	61,243	428,708	61,243	367,465
CO-20%	77,951	69	612,437	122,487	489,950	122,487	367,463	122,487	244,976	122,487	122,489
YA-10%	123,182	37	1,089,040	108,904	980,136	108,904	871,232	108,904	762,328	108,904	653,424
YA-20%	123,182	37	1,089,040	217,808	871,232	217,808	653,424	127,808	435,616	217,808	217,808

E Implementation Details

All experiments are conducted on a machine equipped with 4 NVIDIA RTX 3090Ti GPUs, utilizing the PyTorch framework [57]. As the base Knowledge Graph Embedding (KGE) model, we adopt TransE [27], a well-established model for knowledge graph completion, to ensure a consistent baseline for comparison. Our method GraphDPO, and all baselines are implemented using the same KGE model architecture, which is pre-trained on the same original knowledge graph (KG) to ensure fairness in the experimental setup. The embedding size is 200, and the margin γ for KGE pretraining is 8. For all baselines, we use the Adam as the optimizer. We tune the learning rate from the set [1e-3, 5e-3, 1e-2] and explore different batch sizes from [512, 1024] to identify the best configuration for convergence. For GraphDPO, we set the regularization parameters $\beta = \lambda_1 = \lambda_2 = \lambda_3 = 1$ based on preliminary experiments, ensuring a balanced trade-off between different loss components. Additionally, we set the hyperparameter $\gamma = 8$ to control the margin in the distance-based loss function. To avoid replaying an excessive number of triples, we limit the replay dataset size $|\mathcal{D}_{replay}|$ to 10% of the full dataset. Finally, all experimental results are reported as the average performance over five independent runs to ensure statistical reliability.

F Limitations

While our method GraphDPO demonstrates strong performance across diverse KGE models and unlearning settings, one limitation is its reliance on pre-defined unlearning rates (e.g., 10% or 20%) when constructing the forgetting datasets. In practical scenarios, the scope and quantity of knowledge to be unlearned may be uncertain, context-dependent, or evolve over time, making fixed-rate strategies less flexible. Future work could explore adaptive mechanisms that estimate forgetting targets dynamically, such as leveraging confidence scores, usage frequency, or external feedback to determine both when and how much to forget in a more data-driven manner.

G Main Results on CO-10%, CO-20%, YA-10%, and YA-20%

Table 5: Main experimental results on CO-10%, CO-20%, YA-10%, and YA-20%. The bold scores indicate the best results of approximate unlearning methods and underlined scores indicate the second best results in M_{Avg} and M_{F1} . All results are the average of 5 runs and are presented in % form. OOM denotes out of memory with 4 NVIDIA RTX 3090Ti GPUs.

	Time 1				Time 2				Time 3				Time 4			
\mathcal{D}_f : CO-10%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
Re-Train	0.134	0.229	0.548	0.362	0.128	0.238	0.555	0.373	0.122	0.249	0.563	0.388	0.142	0.299	0.579	0.444
Fine-Tune	0.141	0.216	0.537	0.345	0.138	0.230	0.546	0.363	0.136	0.249	0.556	0.387	0.132	0.278	0.573	0.421
RL	0.040	0.051	0.506	0.097	0.049	0.058	0.504	0.109	0.053	0.059	0.503	0.111	0.051	0.056	0.502	0.106
Fisher	OOM	OOM	OOM	OOM												
BS	0.025	0.031	0.503	0.061	0.035	0.040	0.503	0.078	0.044	0.049	0.502	0.092	0.045	0.048	0.502	0.092
NG	0.031	0.084	0.526	0.154	0.031	0.075	0.522	0.139	0.028	0.057	0.515	0.107	0.013	0.051	0.519	0.096
SSD	0.114	0.103	0.495	0.184	0.106	0.101	0.498	0.182	0.099	0.078	0.489	0.143	0.079	0.094	0.507	0.170
ADV-IMP	0.091	0.093	0.501	0.169	0.077	0.079	0.501	0.146	0.080	0.070	0.495	0.129	0.049	0.079	0.515	0.146
Schema	0.102	0.097	0.497	0.175	0.091	0.086	0.498	0.158	0.087	0.071	0.492	0.132	0.071	0.061	0.495	0.115
MetaEU	0.111	0.087	0.488	0.158	0.107	0.074	0.484	0.137	0.097	0.086	0.494	0.157	0.084	0.077	0.497	0.142
GraphDPO	0.010	0.110	0.550	0.197	0.042	0.101	0.529	0.183	0.060	0.092	0.516	0.167	0.066	0.108	0.521	0.194
\mathcal{D}_f : CO-20%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
Re-Train	0.129	0.235	0.553	0.370	0.138	0.289	0.576	0.433	0.132	0.391	0.629	0.539	0.089	0.552	0.732	0.688
Fine-Tune	0.132	0.214	0.541	0.344	0.131	0.274	0.572	0.417	0.124	0.348	0.612	0.498	0.118	0.448	0.665	0.594
RL	0.047	0.057	0.505	0.108	0.048	0.054	0.503	0.103	0.050	0.055	0.502	0.103	0.050	0.053	0.502	0.100
Fisher	OOM	OOM	OOM	OOM												
BS	0.040	0.049	0.504	0.094	0.048	0.053	0.503	0.101	0.049	0.053	0.502	0.101	0.049	0.051	0.501	0.098
NG	0.042	0.081	<u>0.519</u>	0.149	0.032	0.071	<u>0.520</u>	0.133	0.027	0.068	<u>0.521</u>	0.128	0.043	0.067	<u>0.512</u>	0.126
SSD	0.115	0.095	0.490	<u>0.171</u>	0.095	0.083	0.494	0.152	0.085	0.081	0.498	0.150	0.073	0.077	0.502	0.142
ADV-IMP	0.111	0.091	0.490	0.166	0.101	0.090	0.495	<u>0.164</u>	0.091	0.081	0.495	0.149	0.084	0.075	0.495	0.138
Schema	0.083	0.077	0.497	0.141	0.085	0.080	0.497	0.146	0.094	0.086	0.496	<u>0.158</u>	0.108	0.116	0.504	0.206
MetaEU	0.100	0.076	0.488	0.141	0.113	0.084	0.485	0.154	0.110	0.080	0.485	0.147	0.093	0.084	0.495	0.154
GraphDPO	0.055	0.098	0.521	0.178	0.057	0.099	0.521	0.179	0.067	0.111	0.522	0.199	0.078	0.171	0.546	0.288
\mathcal{D}_f : YA-10%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
Re-Train	0.091	0.158	0.533	0.269	0.091	0.168	0.539	0.284	0.090	0.181	0.545	0.302	0.091	0.203	0.556	0.332
Fine-Tune	0.079	0.123	0.522	0.217	0.078	0.133	0.527	0.233	0.080	0.144	0.532	0.249	0.080	0.156	0.538	0.267
RL	0.021	0.028	0.504	0.055	0.025	0.030	0.502	0.057	0.025	0.029	0.502	0.056	0.026	0.029	0.501	0.057
Fisher	OOM	OOM	OOM	OOM												
BS	0.011	0.015	0.502	0.029	0.019	0.022	0.502	0.043	0.024	0.027	0.502	0.053	0.024	0.026	0.501	0.051
NG	0.124	0.125	0.501	<u>0.218</u>	0.119	0.119	0.500	<u>0.210</u>	0.073	0.113	<u>0.520</u>	<u>0.201</u>	0.114	0.113	0.499	<u>0.200</u>
SSD	0.060	0.087	<u>0.513</u>	0.159	0.062	0.081	<u>0.510</u>	0.149	0.040	0.079	<u>0.520</u>	0.145	0.057	0.073	<u>0.508</u>	0.136
ADV-IMP	0.114	0.115	0.500	0.203	0.113	0.113	0.500	0.200	0.054	0.053	0.500	0.101	0.045	0.045	0.500	0.086
Schema	0.121	0.106	0.493	0.190	0.121	0.119	0.499	0.209	0.115	0.110	0.498	0.196	0.103	0.109	0.503	0.195
MetaEU	0.130	0.088	0.479	0.160	0.110	0.074	0.482	0.137	0.110	0.068	0.479	0.126	0.104	0.057	0.477	0.108
GraphDPO	0.101	0.130	0.514	0.227	0.096	0.132	0.518	0.231	0.086	0.128	0.521	0.225	0.085	0.124	0.520	0.218
\mathcal{D}_f : YA-20%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
Re-Train	0.092	0.167	0.537	0.282	0.091	0.202	0.555	0.330	0.094	0.268	0.587	0.414	0.097	0.501	0.702	0.644
Fine-Tune	0.079	0.131	0.526	0.229	0.077	0.151	0.537	0.259	0.080	0.183	0.551	0.305	0.086	0.234	0.574	0.372
RL	0.023	0.030	0.503	0.058	0.025	0.029	0.502	0.056	0.024	0.026	0.501	0.051	0.023	0.025	0.501	0.049
Fisher	OOM	OOM	OOM	OOM												
BS	0.020	0.025	0.503	0.049	0.024	0.028	0.502	0.054	0.025	0.027	0.501	0.052	0.025	0.026	0.501	0.051
NG	0.139	0.119	0.490	<u>0.209</u>	0.127	0.136	0.504	<u>0.235</u>	0.124	0.130	0.503	<u>0.226</u>	0.115	0.121	0.503	<u>0.212</u>
SSD	0.057	0.080	<u>0.511</u>	0.147	0.058	0.077	0.509	0.142	0.057	0.077	0.510	0.142	0.054	0.081	<u>0.514</u>	0.150
ADV-IMP	0.076	0.076	0.500	0.141	0.070	0.070	0.500	0.131	0.063	0.064	0.500	0.119	0.047	0.048	0.500	0.091
Schema	0.091	0.102	0.506	0.184	0.087	0.109	<u>0.511</u>	0.195	0.092	0.121	<u>0.515</u>	0.214	0.110	0.097	0.494	0.175
MetaEU	0.097	0.097	0.500	0.175	0.103	0.092	0.495	0.167	0.107	0.102	0.498	0.183	0.118	0.084	0.483	0.153
GraphDPO	0.089	0.122	0.517	0.215	0.096	0.138	0.521	0.239	0.110	0.166	0.528	0.279	0.090	0.126	0.518	0.221

H Ablation Results on CO-20% and YA-20%

Table 6: Ablation experimental results on CO-20% and YA-20%. Replay, Dis, and O-S denote boundary replay, boundary distillation and out-boundary sampling, respectively. All results are the average of 5 runs and are presented in % form.

	Time 1				Time 2				Time 3				Time 4			
\mathcal{D}_f : CO-20%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
GraphDPO	0.055	0.098	0.521	0.178	0.057	0.099	0.521	0.179	0.067	0.111	0.522	0.199	0.078	0.171	0.546	0.288
w/o DPO	0.074	0.091	<u>0.508</u>	0.165	0.081	0.091	0.505	0.165	0.085	0.104	0.510	0.187	0.083	0.151	0.534	0.259
w/o Replay	0.100	0.041	0.470	0.078	0.042	0.055	0.507	0.105	0.061	0.099	<u>0.519</u>	0.180	0.073	0.158	<u>0.542</u>	0.270
w/o Dis	0.052	0.024	0.486	0.046	0.036	0.019	0.492	0.038	0.028	0.018	0.495	0.036	0.026	0.019	0.497	0.038
w/o O-S	0.060	0.061	0.500	0.114	0.068	0.088	<u>0.510</u>	0.161	0.076	0.106	0.515	0.190	0.086	0.167	0.541	0.282
\mathcal{D}_f : YA-20%	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}	$M_f \downarrow$	M_r	M_{Avg}	M_{F1}
GraphDPO	0.089	0.122	0.517	0.215	0.096	0.138	0.521	0.239	0.110	0.166	0.528	0.279	0.090	0.126	0.518	0.221
w/o DPO	0.138	0.048	0.455	<u>0.091</u>	0.147	0.067	0.460	<u>0.124</u>	0.136	0.082	0.473	<u>0.150</u>	0.138	0.118	0.490	<u>0.208</u>
w/o Replay	0.132	0.041	0.455	0.079	0.057	0.050	0.4									