Noisy Positive-Unlabeled Learning with Self-Training for Speculative Knowledge Graph Reasoning

Anonymous ACL submission

Abstract

This paper studies speculative reasoning task on real-world knowledge graphs (KG) that contain both *false negative issue* (i.e., potential true facts being excluded) and *false positive issue* (i.e., unreliable or outdated facts being included). State-of-the-art methods fall short in the speculative reasoning ability, as they assume the correctness of a fact is solely determined by its presence in KG, making them vulnerable to false negative/positive issues. The new reasoning task is formulated as a noisy Positive-Unlabeled learning problem. We propose a variational framework, namely nPUGraph, that jointly estimates the correctness of both collected and uncollected facts (which we call *label posterior*) and updates model parameters during training. The label posterior estimation facilitates speculative reasoning from two perspectives. First, it improves the robustness of a label posterior-aware graph encoder against false positive links. Second, it identifies missing facts to provide high-quality grounds of reasoning. They are unified in a simple yet effective self-training procedure. Empirically, extensive experiments on three benchmark KG and one Twitter dataset with various degrees of false negative/positive cases demonstrate the effectiveness of nPUGraph.

1 Introduction

Knowledge graphs (KG), which store real-world facts in triples (*head entity*, *relation*, *tail entity*), have facilitated a wide spectrum of knowledge-intensive applications (Saxena et al., 2021; Qian et al., 2019; Wang et al., 2018, 2022). Automatically reasoning facts based on observed ones, a.k.a. Knowledge Graph Reasoning (KGR) (Bordes et al., 2013), becomes increasingly vital since it allows for expansion of the existing KG at a low cost.

Numerous efforts have been devoted to KGR task (Bordes et al., 2013; Lin et al., 2015; Trouillon et al., 2017; Sun et al., 2019; Li et al., 2021), which assume the correctness of a fact is solely determined by its presence in KG. They ideally view facts included in KG as positive samples and excluded facts as negative samples. However, most real-world reasoning has to be performed based on sparse and unreliable observations, where there may be true facts excluded or false facts included. Reasoning facts based on sparse and unreliable observations (which we call *speculative KG reasoning*) are still underexplored.

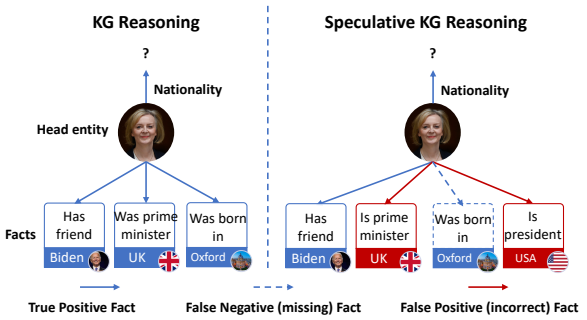


Figure 1: An illustrative example of speculative KG reasoning. Blue solid lines denote the true positive fact, blue dashed lines denote the *false negative* (missing) fact, and red solid lines denote the *false positive* (incorrect) fact. We aim to mitigate the false negative/positive issues to enable speculative KG reasoning.

In this paper, we aim to enable the speculative reasoning ability on real-world KG. The fulfillment of the goal needs to address two commonly existing issues, as shown in Figure 1: 1) **The false negative issue** (i.e., sparse observation): Due to the graph incompleteness, facts excluded from the KG can be used as implicit grounds of reasoning. This is particularly applicable to non-obvious facts. For example, personal information such as the birthplace of politicians may be missing when constructing a political KG, as they are not explicitly stated in the political corpus (Tang et al., 2022). However, it can be critical while reasoning personal facts like nationality. 2) **The false positive issue** (i.e., noisy observation): Facts included in the KG may be unreliable and should not be directly grounded without

inspection. It can happen when relations between entities are incorrectly collected or when facts are extracted from outdated or unreliable sources. For example, Mary Elizabeth is no longer the Prime Minister of the United Kingdom, which may affect the reasoning accuracy of her current workplace. These issues generally affect both one-hop reasoning (Bordes et al., 2013) and multi-hop reasoning (Saxena et al., 2021). The main focus of this paper is investigating the one-hop speculative reasoning task as it lays the basis for complicated multi-hop reasoning capability.

Speculative KG reasoning differs from conventional KG reasoning in that the correctness of each collected/uncollected fact needs to be dynamically estimated as part of the learning process, such that the grounds of reasoning can be accordingly calibrated. Unfortunately, most existing work, if not all, lacks such inspection capability. Knowledge graph embedding methods (Bordes et al., 2013; Lin et al., 2015; Yang et al., 2014; Trouillon et al., 2017; Sun et al., 2019) and graph neural network (GNN) methods (Schlichtkrull et al., 2017; Dettmers et al., 2018; Nguyen et al., 2018; Vashishth et al., 2020; Li et al., 2021) can easily overfit the false negative/positive cases because of their training objective that ranks the collected facts higher than other uncollected facts in terms of plausibility. Recent attempts on uncertain KG (Chen et al., 2019; Kertkeidkachorn et al., 2019) measure the uncertainty scores for facts, which can be utilized to detect false negative/positive samples. However, they explicitly require the ground truth uncertainty scores as supervision for reasoning model training, which are usually unavailable in practice.

Motivated by these observations, we formulate the speculative KG reasoning task as a noisy Positive-Unlabeled learning problem. The facts contained in the KG are seen as noisy positive samples with a certain level of label noise, and the facts excluded from the KG are treated as unlabeled samples, which include both negative ones and possible factual ones. Instead of determining the correctness of facts before training the reasoning model without inspection, we learn the two perspectives in an end-to-end training process. To this end, we propose nPUGraph, a novel variational framework that regards the underlying correctness of collected/uncollected facts in the KG as latent variables for the reasoning process. We jointly update model parameters and estimate the posterior likelihood of the correctness of each collected/uncollected fact (referred to as *label posterior*), through maximizing a theoretical lower bound of the log-likelihood of each fact being collected or uncollected.

The estimated label posterior further facilitates the speculative KG reasoning from two aspects: 1) It removes false positive facts contained in KG and improves the representation quality. We accordingly propose a label posterior-aware encoder to incorporate information only from entity neighbors induced by facts with a high posterior probability, under the assumption that the true positive facts from the collected facts provide more reliable information for reasoning. 2) It complements the grounds of reasoning by selecting missing but possibly plausible facts with high label posterior, which are iteratively added to acquire more informative samples for model training. These two procedures are ultimately unified in a simple yet effective self-training strategy that alternates between the *data sampling based on latest label posteriors* and the *model training based on latest data samples*. Empirically, nPUGraph outperforms eleven state-of-the-art baselines on three benchmark KG data and one Twitter data we collected by large margins. Additionally, its robustness is demonstrated in speculative reasoning on data with multiple ratios of false negative/positive cases.

Our contributions are summarized as follows: (1) We open up a practical but underexplored problem space of speculative KG reasoning, and formulate it as a noisy Positive-Unlabeled learning task; (2) We take the first step in tackling this problem by proposing a variational framework nPUGraph to jointly optimize reasoning model parameters and estimate fact label posteriors; (3) We propose a simple yet effective self-training strategy for nPUGraph to simultaneously deal with false negative/positive issues; (4) We perform extensive evaluations to verify the effectiveness of nPUGraph on both benchmark KG and Twitter interaction data with a wide range of data perturbations.

2 Preliminaries

2.1 Speculative Knowledge Graph Reasoning

A knowledge graph (KG) is denoted as $\mathcal{G} = \{(e_h, r, e_t)\} \subseteq \mathcal{S}$, where $\mathcal{S} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ denotes triple space, $\mathcal{E}$ denotes the entity set, $\mathcal{R}$ denotes the relation set. Each triple $s = (e_h, r, e_t)$ refers to that a head entity $e_h \in \mathcal{E}$ has a relation $r \in \mathcal{R}$

with a tail entity $e_t \in \mathcal{E}$. The goal of speculative KG reasoning is to infer the most plausible triple for each incomplete triple $(e_h, r, e_?)$ or $(e_?, r, e_t)$ given by sparse and unreliable observations in $\mathcal{G}$. In addition, it requires correctness estimation for each potential fact collected or uncollected by $\mathcal{G}$.

2.2 Noisy Positive-Unlabeled Learning

Positive-Unlabeled (PU) learning is a learning paradigm for training a model when only positive and unlabeled data is available (Plessis et al., 2015). We formulate the speculative KG reasoning task as a noisy Positive-Unlabeled learning problem, where the positive set contains potentially label noise from false facts (Jain et al., 2016).

PU Triple Distribution. For the speculative KG reasoning task, we aim to learn a binary classifier that maps a triple space $\mathcal{S}$ to a label space $\mathcal{Y} = \{0, 1\}$. Data are split as labeled (collected)[1] triples $s^l \in \mathcal{S}^L$ and unlabeled (uncollected) triples $s^u \in \mathcal{S}^U$. The labeled triples are considered noisy positive samples with a certain level of label noise. The distribution of labeled triples can be represented as follows:

$$s^l \sim \beta\phi_1^l(s^l) + (1-\beta)\phi_0^l(s^l), \tag{1}$$

where ϕ_y^l denotes the probability of being collected over triple space $\mathcal{S}$ for the positive class ($y = 1$) and negative class ($y = 0$), and $\beta \in [0, 1)$ denotes the proportion of true positive samples in labeled data. Unlabeled triples include both negative samples and possible factual samples. The distribution of unlabeled samples can be represented as follows:

$$s^u \sim \alpha\phi_1^u(s^u) + (1-\alpha)\phi_0^u(s^u), \tag{2}$$

where $\phi_y^u = 1 - \phi_y^l$ denotes the probability of being uncollected, $\alpha \in [0, 1)$ denotes the positive class prior, i.e., the proportion of positive samples in unlabeled data.

PU Triple Construction. We then discuss the construction of $\mathcal{S}^L$ and $\mathcal{S}^U$ based on the collected KG $\mathcal{G}$. Triples in $\mathcal{G}$ naturally serve as labeled samples with a ratio of noise, i.e., $\mathcal{S}^L = \mathcal{G}$. For unlabeled set $\mathcal{S}^U$, However, directly using $\mathcal{S} \setminus \mathcal{G}$ as the unlabeled set $\mathcal{S}^U$ would result in too many unlabeled samples for training due to the large number of possible triples in triple space $\mathcal{S}$. Following (Tang et al., 2022), we construct $\mathcal{S}^U$ as follows: For each labeled triple $s_i^l = (e_h, r, e_t)$, we construct K unlabeled triples s_{ik}^u by replacing the head and tail respectively with other entities: $s_{ik}^u = (e_h, r, e_k^-)$ or (e_k^-, r, e_t), where e_k^- is the selected entity that ensures $s_{ik}^u \notin \mathcal{S}^L$. Initially, the construction can be randomized. During the training process, it is further improved by selecting unlabeled samples with high label posterior in a self-training scheme, so as to cover positive samples in the unlabeled set to the greatest extent.

[1] In this paper, we interchangeably use the term *labeled/unlabeled* and *collected/uncollected* with no distinction.

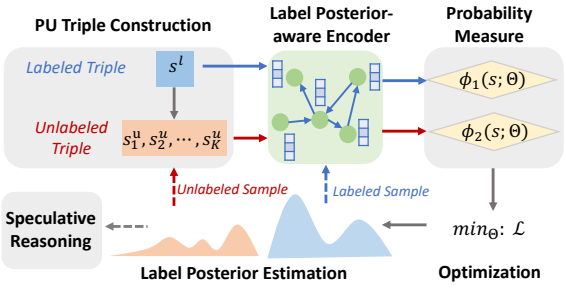


Figure 2: nPUGraph overview. It jointly optimizes parameters and estimates label posterior, to detect false negative/positive cases for the encoder and self-training.

3 Methodology

3.1 Overview

Our approach views underlying triple labels (positive/negative) as latent variables, influencing the collection probability. Unlike the common objective of reasoning training that ranks the plausibility of the collected triples higher than uncollected ones, we instead maximize the data likelihood of each potential triple being collected or not. To this end, as shown in Figure 2, we propose nPUGraph framework to jointly optimize parameters and infer the label posterior. During the training process, the latest label posterior estimation can be utilized by a label posterior-aware encoder, which improves the quality of representation learning by only integrating information from the entity neighbors induced by true facts. Finally, a simple yet effective self-training strategy based on label posterior is proposed, which can dynamically update neighbor sets for the encoder and sample unlabeled triplets to cover positive samples in the unlabeled set to the greatest extent for model training.

The remaining of this section is structured as follows: Section 3.2 first formalizes the learning objective and the variational framework for likelihood maximization. Section 3.3 details the label posterior-aware encoder for representation learning, followed by Section 3.4 that introduces the

self-training strategy.

3.2 Noisy PU Learning on KG

Due to the false negative/positive issues, the correctness of a fact (y) is not solely determined by its presence in a knowledge graph. nPUGraph addresses the issue by treating the underlying label as a latent variable that influences the probability of being collected or not. We, therefore, set maximizing the data collection likelihood as our objective. In such a learning paradigm, the assumptions that collected triples are correct $p(y=1|s^l)=1$ and uncollected triples are incorrect $p(y=0|s^u)=1$ are removed. We aim to train a model on labeled triples $\mathcal{S}^L$ and unlabeled ones $\mathcal{S}^U$, and infer the label posterior $p(y|s^u)$ and $p(y|s^l)$ at the same time by data likelihood maximization. The latest label posterior can help to detect false negative/positive cases during model training.

We first derive our training objective. To be more formal, the log-likelihood of each potential fact being collected or not is lower bounded by Eq. (3), which is given by Theorem 1.

Theorem 1. *The log-likelihood of the complete data* $\log p(\mathbf{S})$ *is lower bounded as follows:*

$$\begin{aligned}
&\log p(\mathbf{S}) \geq \mathop{\mathbb{E}}_{q(\mathbf{Y})}\left[\log p(\mathbf{S}|\mathbf{Y})\right] - \mathbb{KL}(q(\mathbf{Y})\|p(\mathbf{Y})) \\
&= \mathop{\mathbb{E}}_{s^l\in\mathcal{S}^L}\left[w^l\log[\phi_1^l(s^l)] + (1-w^l)\log[\phi_0^l(s^l)]\right] \\
&+ \mathop{\mathbb{E}}_{s^u\in\mathcal{S}^U}\left[(w^u\log[\phi_1^u(s^u)] + (1-w^u)\log[\phi_0^u(s^u)]\right] \\
&- \mathbb{KL}(\mathbf{W}^U\|\tilde{\mathbf{W}}^U) - \mathbb{KL}(\mathbf{W}^L\|\tilde{\mathbf{W}}^L) - \frac{\|\mathbf{W}^L\|_1}{|\mathcal{S}^L|} - \frac{\|\mathbf{W}^U\|_1}{|\mathcal{S}^U|},
\end{aligned} \tag{3}$$

where $\mathbf{S}$ *denotes all labeled/unlabeled triples,* $\mathbf{Y}$ *is the corresponding latent variable indicating the positive/negative labels for triples,* $\mathbf{W}^U=\{w_i^u\}$ *denotes the point-wise probability for the uncollected triples being positive,* $\mathbf{W}^L=\{w_i^l\}$ *denotes the probability for the collected triples being positive.* $\tilde{\mathbf{W}}^U$ *and* $\tilde{\mathbf{W}}^U$ *are the approximation of the collection probability for uncollected/collected triples respectively, produced by nPUGraph based on the latest parameters.*

Proof. Refer to Appendix A.1 for proof. □

We treat label $\mathbf{Y}$ as a latent variable and derive the lower bound for the log-likelihood, which is influenced by the prior knowledge of positive class prior α and true positive ratio β. Thus, maximizing the lower bound can jointly optimize model parameters and infer the posterior label distribution, $\mathbf{W}^U$ and $\mathbf{W}^L$. Such a learning process enables us to avoid false negative/positive issues during model training since it considers ϕ_0^l (one negative triple is collected) and ϕ_1^u (one positive triple is missing) as non-zero probability, which are determined by the latest label posterior during model training.

Probability Measure. We then specify the probability measures for positive/negative triples being collected, i.e., $\phi_1^l(\cdot)$ and $\phi_0^l(\cdot)$ ($\phi_y^u(\cdot)=1-\phi_y^l(\cdot)$ for $y=1/0$). To better connect to other methods utilizing score functions for KGR, we hereby utilize the sigmoid function $\sigma(\cdot)$ to directly transform the score function $\psi(s;\Theta)$, parameterized by model parameters Θ, to probability:

$$\phi_1^l(s)=\sigma(\psi_1(s;\Theta)),\quad \phi_0^l(s)=\sigma(\psi_0(s;\Theta)), \tag{4}$$

we hereby utilize two score functions $\psi_1(s;\Theta)$ and $\psi_0(s;\Theta)$ to measure the positive/negative triples being collected, as the influencing factors based on triple information can be different. We utilize two neural networks to approximate the probability measure, which will be detailed in Section 3.3.

Since we aim to detect the potential existence of positive triples in an unlabeled set, it is unnecessary to push the collection probability of all uncollected triple $\phi_y^l(s^u)$ to 0 ($\phi_y^u(s^u)$ to 1). A loose constraint is that we force the uncollection probability of a collected triple s^l lower than its corresponding uncollected triples s^u: $\phi_y^u(s^l)<\phi_y^u(s^u)$. Therefore, we adopt the pair-wise ranking measure $\phi_y^\star(s^l,s^u)$ to replace $\phi_y^u(s^u)$ as follows:

$$\phi_y^u(s^u)\rightarrow\phi_y^\star(s^l,s^u)=\sigma(\psi_y(s^u;\Theta)-\psi_y(s^l;\Theta)). \tag{5}$$

Maximum Probability Training. We then derive the training objective based on Eq. (3) The first part of Eq. (3) measures the probability of data being collected/uncollected. Concretely, given each collected triple $s_i^l\in\mathcal{S}^L$ and its corresponding K uncollected triples $s_{ik}^u\in\mathcal{S}^U$, We denote the loss function measuring the probability as $\mathcal{L}_{triple}$:

$$\begin{aligned}
\mathcal{L}_{triple} = -\frac{1}{K|\mathcal{S}^L|}\sum_i\sum_k(&w_i^l\log[\phi_1^l(s_i^l)] \\
&+(1-w_i^l)\log[\phi_0^l(s_i^l)] + w_{ik}^u\log[\phi_1^\star(s_i^l,s_{ik}^u)] \\
&+(1-w_{ik}^u)\log[\phi_0^\star(s_i^l,s_{ik}^u)]),
\end{aligned} \tag{6}$$

where w_i^l denotes the point-wise probability for the collected triple s_i^l being positive, w_{ik}^u denotes the probability for the uncollected triple s_{ik}^u being

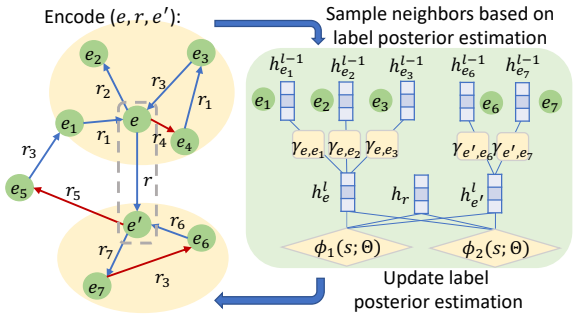


Figure 3: The label posterior-aware encoder. Red line denotes the detected false positive facts based on posterior, which are excluded during neighbor sampling.

positive. Based on the definition, the posterior probability of each collected/uncollected triple being positive can be computed as:

$$\tilde{w}_i^l = \frac{\beta\phi_1^l(s_i^l)}{\beta\phi_1^l(s_i^l) + (1-\beta)\phi_0^l(s_i^l)}, \tag{7}$$

$$\tilde{w}_{ik}^u = \frac{\alpha\phi_1^u(s_{ik}^u)}{\alpha\phi_1^u(s_{ik}^u) + (1-\alpha)\phi_0^u(s_{ik}^u)}. \tag{8}$$

To increase model expression ability, instead of forcing $\mathbf{W}^L = \tilde{\mathbf{W}}^L$ and $\mathbf{W}^U = \tilde{\mathbf{W}}^U$, we set $\mathbf{W}^L$ and $\mathbf{W}^U$ as free parameters and utilize the term $\mathcal{L}_{KL} = \mathbb{KL}(\mathbf{W}^L \| \tilde{\mathbf{W}}^L) + \mathbb{KL}(\mathbf{W}^U \| \tilde{\mathbf{W}}^U)$ to regularize the difference. Finally, based on Eq. (3), the training objective is formalized as follows:

$$\min_{\Theta} \mathcal{L} = \min_{\Theta} \mathcal{L}_{triple} + \mathcal{L}_{KL} + \mathcal{L}_{reg}, \tag{9}$$

where $\mathcal{L}_{reg} = \|\mathbf{W}^L\|_1 + \|\mathbf{W}^U\|_1$ can be viewed as a normalization term. Considering the sparsity property of real-world graphs, $\mathcal{L}_{reg}$ penalizes the posterior estimation that there are too many true positive facts on KG.

3.3 Label Posterior-aware Encoder

We then introduce the encoder and the score functions $\psi_1(s;\Theta)$ and $\psi_0(s;\Theta)$ to measure the probability of positive/negative triples being collected, as shown in Figure 3. Recent work (Schlichtkrull et al., 2017; Dettmers et al., 2018; Nguyen et al., 2018; Vashishth et al., 2020) has shown that integrating information from neighbors to represent entities engenders better reasoning performance. However, the message-passing mechanism is vulnerable to the false positive issue, as noise can be integrated via a link induced by a false positive fact. In light of this, we propose a label posterior-aware encoder to improve the quality of representations.

We represent each entity $e \in \mathcal{E}$ and each relation $r \in \mathcal{R}$ into a d-dimensional latent space: $\mathbf{h}_e, \mathbf{h}_r \in \mathbb{R}^d$. To encode more information in $\mathbf{h}_e$, we first construct a neighbor set $\mathcal{N}_e$ induced by the positive facts related to entity e. The latest label posterior for collected facts $\tilde{\mathbf{W}}^L$ naturally serves this purpose, as it indicates the underlying correctness for each collected fact.

Therefore, for each entity e, we first sort the related facts by label posterior $\tilde{\mathbf{W}}^L$ and construct the neighbor set $\mathcal{N}_e(\tilde{\mathbf{W}}^L) = \{(e_i, r_i)\}$ from the top facts. Then the encoder attentively aggregates information from the collected neighbors, where the attention weights take neighbor features, relation features into account. Specifically:

$$\mathbf{h}_e^l = \mathbf{h}_e^{l-1} + \sigma\left(\sum_{(e_i,r_i)\in\mathcal{N}_e(\tilde{\mathbf{W}}^L)} \gamma_{e,e_i}^l \left(\mathbf{h}_{e_i}^{l-1}\mathbf{M}\right)\right), \tag{10}$$

where l denotes the layer number, $\sigma(\cdot)$ denotes the activation function, γ_{e,e_i}^l denotes the attention weight of entity e_i to the represented entity e, and $\mathbf{M}$ is the trainable transformation matrix. The attention weight γ_{e,e_i}^l is supposed to be aware of entity feature and topology feature induced by relations. We design the attention weight γ_{e,e_i}^l as follows:

$$\gamma_{e,e_i}^l = \frac{\exp(q_{e,e_i}^l)}{\sum_{\mathcal{N}_e(\tilde{\mathbf{W}}^L)} \exp(q_{e,e_k}^l)}, \quad q_{e,e_k}^l = \mathbf{a}\left(\mathbf{h}_e^{l-1} \| \mathbf{h}_{e_k}^{l-1} \| \mathbf{h}_{r_k}\right), \tag{11}$$

where q_{e,e_k}^l measures the pairwise importance from neighbor e_k by considering the entity embedding, neighbor embedding, and relation embedding, $\mathbf{a} \in \mathbb{R}^{3d}$ is a shared parameter in the attention.

To measure the collection probability for positive/negative triples, we utilize two multilayer perception (MLP) to approximate score function $\psi_1(s;\Theta)$ and $\psi_0(s;\Theta)$. Specifically, for each triple $s = (e_h, r, e_t)$:

$$\psi_1(s;\Theta) = \text{MLP}_1(\mathbf{h}_s), \ \psi_0(s;\Theta) = \text{MLP}_0(\mathbf{h}_s), \tag{12}$$

where the MLP input $\mathbf{h}_s = [\mathbf{h}_{e_h}^l \| \mathbf{h}_r \| \mathbf{h}_{e_t}^l]$ concatenates entity embeddings and relation embedding.

3.4 Self-Training Strategy

The latest label posterior $\tilde{\mathbf{W}}^L$ and $\tilde{\mathbf{W}}^U$ is further utilized in a self-training strategy to enhance speculative reasoning. First, the latest posterior estimation $\tilde{\mathbf{W}}^L$ for collected links updates neighbor sets to gradually prevent the encoder effects by false positive links. Moreover, the latest estimation $\tilde{\mathbf{W}}^L$ for uncollected facts enables us to continuously sample unlabeled triplets with high label posterior to cover positive samples in the unlabeled set to

Algorithm 1: Summary of nPUGraph.

Input: The collected triple set $\mathcal{S}^L$.
Output: The model parameter Θ, predicted triples.

Construct the uncollected triple set $\mathcal{S}^U$ randomly;
Initialize the model parameter Θ and the label posterior $\tilde{\mathbf{W}}^L$ and $\tilde{\mathbf{W}}^U$ randomly;
for *each training epoch* **do**
  Construct neighbor set $\mathcal{N}_e(\tilde{\mathbf{W}}^L)$ by $\tilde{\mathbf{W}}^L$;
  Construct uncollected triple set $\mathcal{S}^U$ by $\tilde{\mathbf{W}}^U$;
  for *each collected triple* $s_i^l \in \mathcal{S}^L$ **do**
    Collect unlabeled triples $\{s_{ik}^u\}_{k=1}^K$;
    Calculate $\phi_y^l(s_i^l)$ by Eq. (4);
    Calculate each $\phi_y^\star(s_i^l, s_{ik}^u)$ by Eq. (5);
  end
  Calculate the total loss $\mathcal{L}$ by Eq. (9);
  Optimize model parameter: $\Theta = \Theta - \frac{\partial \mathbf{L}}{\partial \Theta}$;
  Update label posterior $\tilde{\mathbf{W}}^L$ and $\tilde{\mathbf{W}}^U$;
end

the greatest extent. For each labeled triple $s_i^l = (e_h, r, e_t)$, we construct K unlabeled triples s_{ik}^u by replacing the head and tail respectively with other entities: $s_{ik}^u = (e_h, r, e_k^-)$ or (e_k^-, r, e_t), where e_k^- is the selected entity that ensures $s_{ik}^u \notin \mathcal{S}^L$. Such selection is performed by ranking the corresponding label posterior $\tilde{w}_{ik}^u$. The updates of neighbor sets and unlabeled triples based on label posterior are nested with parameter optimization during model training alternatively. The training of nPUGraph is summarized in Algorithm 1.

4 Experiment

4.1 Experimental Setup

Dataset. We evaluate nPUGraph mainly on three benchmark datasets: FB15K (Bordes et al., 2013), FB15k-237 (Toutanova et al., 2015), and WN18 (Bordes et al., 2013) and one Twitter data we collected, which describes user interaction information towards tweets and hashtags. Table 3 in Appendix A.2 summarizes the dataset statistics.

To better fit the real scenario for speculative reasoning, we randomly modify links on KG to simulate more false negative/positive cases. We modify a specific amount of positive/negative links (the ratio of the modified links is defined as perturbation rate, i.e., *ptb_rate*) by flipping. 90% of them are the removed positive links to simulate false negative cases and the remaining 10% are the added negative links to simulate false positive cases. More details about datasets and the data perturbation process can be found in Appendix A.2.

Baselines. We compare to eleven state-of-the-art baselines: 1) KG embedding methods: **TransE** (Bordes et al., 2013), **TransR** (Lin et al., 2015), **DistMult** (Yang et al., 2014), **ComplEX** (Trouillon et al., 2017), and **RotatE** (Sun et al., 2019); 2) GNN methods on KG: **RGCN** (Schlichtkrull et al., 2017) and **CompGCN** (Vashishth et al., 2020); 3) Uncertain KG reasoning: **UKGE** (Chen et al., 2019); 4) Negative sampling methods: **NSCaching** (Zhang et al., 2019) and **SANS** (Ahrabian et al., 2020); 5) PU learning on KG: **PUDA** (Tang et al., 2022). More details can be found in Appendix A.3.

Evaluation and Implementation. For each $(e_h, r, e_?)$ or $(e_?, r, e_t)$, we rank all entities at the missing position in triples, and adopt filtered mean reciprocal rank (*MRR*) and filtered Hits at $\{1, 3, 10\}$ as evaluation metrics (Bordes et al., 2013). More implementation details of baselines and nPUGraph can be found in Appendix A.4.

4.2 Main Results

We first discuss the model performance on noisy and incomplete graphs, with *ptb_rate = 0.3*, as shown in Table 1. nPUGraph achieves consistently better results than all baseline models, with 10.3% relative improvement on average. Specifically, conventional KGE and GNN-based methods produce unsatisfying performance, as they ignore the false negative/positive issues during model training. In some cases, GNN-based ways are worse, as the message-passing mechanism is more vulnerable to false positive links. As expected, the performance of uncertain knowledge graph embedding model (Chen et al., 2019) is much worse when there are no available uncertainty scores for model training. SANS and PUDA generate competitive results in some cases, as their negative sampling strategy and PU learning objective can respectively mitigate the false negative/positive issues to some extent. Table 1 demonstrates the superiority of nPUGraph which addresses the false negative/positive issues simultaneously. For space limitations, we report and discuss the model performance on Twitter data in Table 4 in Appendix A.5.1.

4.3 Experiments under Various Degrees of Noise and Incompleteness

We investigate the performance of baseline models and nPUGraph under different degrees of noise and incompleteness. Figure 4 reports the performance under various *ptb_rate* , from 0.1 to 0.7, where higher *ptb_rate* means more links are perturbed as false positive/negative cases. Full re-

Table 1: Overall performance on noisy and incomplete graphs, with *ptb_rate = 0.3*. Average results on 5 independent runs are reported. $*$ indicates the statistically significant results over baselines, with p-value < 0.01. The best results are in boldface, and the strongest baseline performance is underlined.

Dataset	FB15K				FB15K-237				WN18			
Metrics	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**
Knowledge graph embedding methods												
TransE	0.336	0.603	0.425	0.189	0.196	0.394	0.236	0.094	0.229	0.481	0.416	0.030
TransR	0.314	0.579	0.397	0.170	0.184	0.359	0.211	0.098	0.229	0.480	0.408	0.035
DistMult	0.408	0.627	0.463	0.296	0.240	0.407	0.262	0.158	0.397	0.518	0.453	0.320
ComplEx	0.396	0.616	0.451	0.284	0.238	0.411	0.262	0.154	0.448	0.526	0.475	0.403
RotatE	0.431	0.636	0.489	0.323	0.255	0.433	0.280	0.169	0.446	0.524	0.474	0.400
Graph neural network methods on KG												
RGCN	0.154	0.307	0.164	0.078	0.141	0.276	0.145	0.075	0.362	0.464	0.412	0.300
CompGCN	0.409	0.631	0.465	0.294	0.253	0.422	0.275	0.171	0.445	0.522	0.471	0.400
Uncertain knowledge graph embedding method												
UKGE	0.311	0.556	0.337	0.189	0.172	0.233	0.128	0.081	0.241	0.447	0.309	0.119
Negative sampling methods												
NSCaching	0.371	0.576	0.424	0.265	0.190	0.329	0.208	0.121	0.306	0.401	0.334	0.255
SANS	0.372	0.599	0.434	0.252	0.243	0.416	0.267	0.158	0.453	0.528	0.479	0.409
Positive-Unlabeled learning methods on KG												
PUDA	0.403	0.623	0.458	0.291	0.234	0.394	0.255	0.156	0.382	0.499	0.444	0.306
nPUGraph	**0.486***	**0.718***	**0.534***	**0.342***	**0.287***	**0.481***	**0.315***	**0.191***	**0.493***	**0.582***	**0.519***	**0.442***
Gains (%)	*12.7*	*12.8*	*9.2*	*5.9*	*12.6*	*11.2*	*12.5*	*11.4*	*8.9*	*10.3*	*8.3*	*8.0*

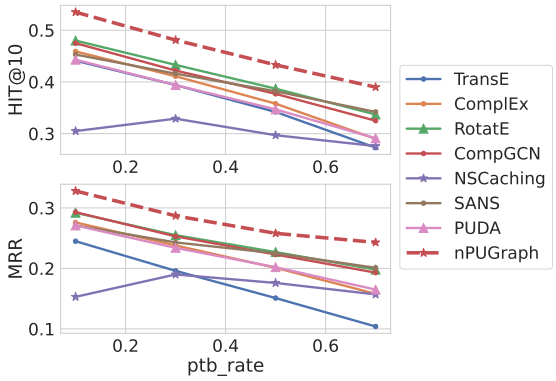


Figure 4: Performance with respect to various *ptb_rate* , i.e., different degrees of noise and incompleteness, on *FB15K-237*. nPUGraph exhibits impressive robustness against false negative/positive issues.

sults are included in Appendix A.5.2. The performance degrades as the *ptb_rate* increases for all in most cases, demonstrating that the false negative/positive issues significantly affect the reasoning performance. However, nPUGraph manages to achieve the best performance in all cases. Notably, the relative improvements are more significant under higher *ptb_rate* .

4.4 Model Analysis

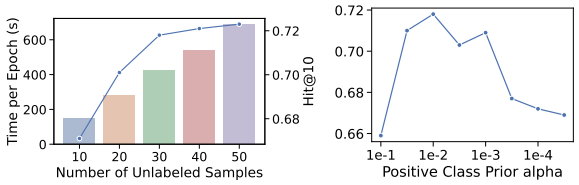
Ablation Study. We evaluate performance improvements brought by the nPUGraph framework by following ablations: 1) **nPUGraph w/o nPU** is trained without the noisy Positive-Unlabeled framework, which instead utilizes the margin loss for model training; 2) **nPUGraph w/o LP-Encoder** eliminates the label posterior-aware encoder (LP-Encoder), which aggregates information from all neighbors instead of the sampled neighbors; 3) **nPUGraph w/o Self-Training** is trained without the proposed self-training algorithm.

Table 2: Ablation Studies.

Dataset	FB15K		FB15K-237		Gains
Ablations	**MRR**	**H@10**	**MRR**	**H@10**	**%**
nPUGraph w/o nPU	0.401	0.619	0.230	0.407	-20.0
nPUGraph w/o LP-Encoder	0.457	0.681	0.261	0.459	-6.6
nPUGraph w/o Self-Training	0.471	0.704	0.276	0.461	-3.4
nPUGraph	0.486	0.718	0.287	0.481	-

We report *MRR* and *Hit@10* over FB15K and FB15K-237 data, as shown in Table 2. As we can see, training the encoder without the proposed noisy Positive-Unlabeled framework will cause the performance drop, as this variant ignores the false negative/positive issues. Removing the label posterior-based neighbor sampling in the encoder will also cause performance degradation, as the information aggregation no longer distinguishes between true and false links. Such a variant can be easily influenced by the existence of false positive facts. Moreover, the last ablation result shows if the training process is further equipped with the self-training strategy, the performance will be enhanced, which verifies its effectiveness to select informative unlabeled samples for model training.

The Effect of PU Triple Construction. We then investigate the effect of PU Triple Construction on model performance, by varying different sizes of unlabeled samples from 10 to 50. Figure 5a shows that the performance improves as the num-



(a) #unlabeled samples. (b) Class prior α.

Figure 5: Model analysis w.r.t. the number of unlabeled samples and positive class prior α.

ber of unlabeled samples increases. Because more unlabeled samples can cover more false negative cases for model training. The training time grows linearly.

The Effect of Positive Class Prior α. The positive class prior α and true positive ratio β are two important hyperparameters. While β has a clear definition from real-world data, the specific value of α is unknown in advance. Figure 5b shows the model performance w.r.t. different values of α by grid search. Reasoning performance fluctuates a bit with different values of α since incorrect prior knowledge of α can bias the label posterior estimation and thus hurt the performance.

5 Related Work

Knowledge Graph Reasoning. Knowledge graph reasoning (KGR) aims to predict missing facts to automatically complete KG, including one-hop reasoning (Bordes et al., 2013) and multi-hop reasoning (Saxena et al., 2021). To set the scope, we primarily focus on one-hop reasoning and are particularly interested in predicting missing entities in a partial fact. Knowledge graph embeddings achieve state-of-the-art performance (Bordes et al., 2013; Lin et al., 2015; Yang et al., 2014; Trouillon et al., 2017; Sun et al., 2019; García-Durán et al., 2018). Recently, graph neural networks (GNN) have been incorporated to enhance representation learning by aggregating neighborhood information (Schlichtkrull et al., 2017; Dettmers et al., 2018; Nguyen et al., 2018; Vashishth et al., 2020; Li et al., 2021). However, most approaches significantly degrade when KG are largely incomplete and contain certain errors (Pujara et al., 2017), as they ignore the false negative/positive issues. Recent attempts on uncertain KG (Chen et al., 2019; Kertkeidkachorn et al., 2019; Chen et al., 2021) measure the uncertainty score for facts, which can detect false negative/positive samples. However, they explicitly require the ground truth uncertainty scores for model training, which are usually unavailable in practice. Various negative sampling strategies have been explored to sample informative negative triples to facilitate model training (Cai and Wang, 2018; Zhang et al., 2019; Ahrabian et al., 2020). However, they cannot detect false negative/positive facts. We aim to mitigate the false negative/positive issues and enable the automatic detection of false negative/positive facts during model training.

Positive-Unlabeled Learning. Positive-Unlabeled (PU) learning is a learning paradigm for training a model that only has access to positive and unlabeled data, where unlabeled data includes both positive and negative samples (Plessis et al., 2015; Bekker and Davis, 2018). PU learning roughly includes (1) two-step solutions (He et al., 2018; Jain et al., 2016); (2) methods that consider the unlabeled samples as negative samples with label noise (Shi et al., 2018); (3) unbiased risk estimation methods (Plessis et al., 2015; Tang et al., 2022). Recent work further studies the setting that there exists label noise in the observed positive samples (Jain et al., 2016). We formulate the KGR task on noisy and incomplete KG as a noisy Positive-Unlabeled learning problem and propose a variational framework for it, which relates to two-step solutions and unbiased risk estimation methods.

6 Conclusion

We studied speculative KG reasoning based on sparse and unreliable observations, which contains both *false negative issue* and *false positive issue*. We formulated the task as a noisy Positive-Unlabeled learning problem and proposed a variational framework nPUGraph to jointly update model parameters and estimate the posterior likelihood of collected/uncollected facts being true or false, where the underlying correctness is viewed as latent variables. During the training process, a label posterior-aware encoder and a self-training strategy were proposed to further address the false positive/negative issues. We found label posterior estimation plays an important role in moving toward speculative KG reasoning in reality, and the estimation can be fulfilled by optimizing an alternative objective without additional cost. Extensive experiments verified the effectiveness of nPUGraph on both benchmark KGs and Twitter interaction data with various degrees of data perturbations.

7 Limitations

There are certain limitations that can be concerned for further improvements. First, the posterior inference relies on the prior estimation of positive class prior α and true positive ratio β. Our experiments show that a data-driven estimation based on end-to-end model training produces worse results than a hyperparameter grid search. An automatic prior estimation is desirable for real-world applications. Moreover, in nPUGraph, we approximate the probability of negative/positive facts being collected/uncollected via neural networks, which lacks a degree of interpretability. In the future, we plan to utilize a more explainable random process depending on entity/relation features to model the collection probability distribution.

8 Ethical Impact

nPUGraph neither introduces any social/ethical bias to the model nor amplifies any bias in data. Benchmark KG are publicly available. For Twitter interaction data, we mask all identity and privacy information for users, where only information related to user interactions with tweets and hashtags is presented. Our model is built upon public libraries in PyTorch. We do not foresee any direct social consequences or ethical issues.

References

Kian Ahrabian, Aarash Feizi, Yasmin Salehi, William L. Hamilton, and Avishek Joey Bose. 2020. Structure aware negative sampling in knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6093–6101, Online. Association for Computational Linguistics.

Jessa Bekker and Jesse Davis. 2018. Learning from positive and unlabeled data: A survey. *CoRR*, abs/1811.04820.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*.

Liwei Cai and William Yang Wang. 2018. KBGAN: Adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1470–1480, New Orleans, Louisiana. Association for Computational Linguistics.

Xuelu Chen, Michael Boratko, Muhao Chen, Shib Sankar Dasgupta, Xiang Lorraine Li, and Andrew McCallum. 2021. Probabilistic box embeddings for uncertain knowledge graph reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xuelu Chen, Muhao Chen, Weijia Shi, Yizhou Sun, and Carlo Zaniolo. 2019. Embedding uncertain knowledge graphs. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Fengxiang He, Tongliang Liu, Geoffrey I. Webb, and Dacheng Tao. 2018. Instance-dependent PU learning by bayesian optimal relabeling. *CoRR*, abs/1808.02180.

Shantanu Jain, Martha White, and Predrag Radivojac. 2016. Estimating the class prior and posterior from noisy positives and unlabeled data. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2693–2701, Red Hook, NY, USA. Curran Associates Inc.

Natthawut Kertkeidkachorn, Xin Liu, and Ryutaro Ichise. 2019. Gtranse: Generalizing translation-based model on uncertain knowledge graph embedding. In *Advances in Artificial Intelligence - Selected Papers from the Annual Conference of Japanese Society of Artificial Intelligence (JSAI 2019), Niigata, Japan, 4-7 June 2019*, volume 1128 of *Advances in Intelligent Systems and Computing*, pages 170–178.

Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. Temporal knowledge graph reasoning based on evolutional representation learning. In *SIGIR*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI'15*.

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 *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 327–333.

Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. 2015. Convex formulation for learning from positive and unlabeled data. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1386–1394, Lille, France.

Jay Pujara, Eriq Augustine, and Lise Getoor. 2017. Sparsity and noise: Where knowledge graph embeddings fall short. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1751–1756, Copenhagen, Denmark. Association for Computational Linguistics.

Jianwei Qian, Xiang-Yang Li, Chunhong Zhang, Linlin Chen, Taeho Jung, and Junze Han. 2019. Social network de-anonymization and privacy inference with knowledge graph model. *IEEE Transactions on Dependable and Secure Computing*, pages 679–692.

Apoorv Saxena, Soumen Chakrabarti, and Partha Talukdar. 2021. Question answering over temporal knowledge graphs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607.

Hong Shi, Shaojun Pan, Jian Yang, and Chen Gong. 2018. Positive and unlabeled learning via loss decomposition and centroid estimation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 2689–2695. AAAI Press.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Zhenwei Tang, Shichao Pei, Zhao Zhang, Yongchun Zhu, Fuzhen Zhuang, Robert Hoehndorf, and Xiangliang Zhang. 2022. Positive-unlabeled learning with adversarial data augmentation for knowledge graph completion. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2248–2254. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal. Association for Computational Linguistics.

Théo Trouillon, Christopher R. Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2017. Knowledge graph completion via complex tensor factorization. *J. Mach. Learn. Res.*

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM '18*.

Ruijie Wang, Zheng Li, Danqing Zhang, Qingyu Yin, Tong Zhao, Bing Yin, and Tarek Abdelzaher. 2022. Rete: Retrieval-enhanced temporal event forecasting on unified query product evolutionary graph. In *The Web Conference*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Yongqi Zhang, Quanming Yao, Yingxia Shao, and Lei Chen. 2019. Nscaching: Simple and efficient negative sampling for knowledge graph embedding. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 614–625. IEEE.

A Appendix

A.1 Theorem Proof

Theorem 2. *The log-likelihood of the complete data* $\log p(\mathbf{S})$ *is lower bounded as follows:*

$$\begin{aligned}
&\log p(\mathbf{S}) \geq \mathop{\mathbb{E}}_{q(\mathbf{Y})}\left[\log p(\mathbf{S}|\mathbf{Y})\right] - \mathbb{KL}(q(\mathbf{Y})\|p(\mathbf{Y})) \\
&= \mathop{\mathbb{E}}_{s^l\in\mathcal{S}^L}\left[w^l \log[\phi_1^l(s^l)] + (1-w^l)\log[\phi_0^l(s^l)]\right] \\
&+ \mathop{\mathbb{E}}_{s^u\in\mathcal{S}^U}\left[(w^u \log[\phi_1^u(s^u)] + (1-w^u)\log[\phi_0^u(s^u)]\right] \\
&- \mathbb{KL}(\mathbf{W}^U\|\tilde{\mathbf{W}}^U) - \mathbb{KL}(\mathbf{W}^L\|\tilde{\mathbf{W}}^L) - \frac{\|\mathbf{W}^L\|_1}{|\mathcal{S}^L|} - \frac{\|\mathbf{W}^U\|_1}{|\mathcal{S}^U|},
\end{aligned} \tag{13}$$

where $\mathbf{S}$ *denotes all labeled/unlabeled triples,* $\mathbf{Y}$ *is the corresponding latent variable indicating the positive/negative labels for triples,* $\mathbf{W}^U = \{w_i^u\}$ *denotes the point-wise probability for the uncollected triples being positive,* $\mathbf{W}^L = \{w_i^l\}$ *denotes the probability for the collected triples being positive.*

Proof. Let $\log p(\mathbf{S})$ denote the log-likelihood of all potential triples being collected in the KG or not, $\mathbf{Y}$ denote the corresponding latent variable indicating the positive/negative labels. We aim to infer the label posterior $p(\mathbf{Y}|\mathbf{S})$, which can be approximated by $q(\mathbf{Y}|\mathbf{S})$. We therefore are interested at the difference between the two, measured by the Kullback–Leibler (KL) divergence as follows:

$$\mathbb{KL}(q(\mathbf{Y}|\mathbf{S})\|p(\mathbf{Y}|\mathbf{S})) = -\mathop{\mathbb{E}}_{q(\mathbf{Y}|\mathbf{S})}\log\left(\frac{p(\mathbf{S}|\mathbf{Y})p(\mathbf{Y})}{q(\mathbf{Y}|\mathbf{S})}\right) + \log p(\mathbf{S}), \tag{14}$$

as KL divergence is positive, we derive the lower bound of the log-likelihood as follows:

$$\begin{aligned}
\log p(\mathbf{S}) &\geq \mathop{\mathbb{E}}_{q(\mathbf{Y}|\mathbf{S})}\log\left(\frac{p(\mathbf{S}|\mathbf{Y})p(\mathbf{Y})}{q(\mathbf{Y}|\mathbf{S})}\right) \\
&\geq \mathop{\mathbb{E}}_{q(\mathbf{Y}|\mathbf{S})}\log p(\mathbf{S}|\mathbf{Y}) - \mathbb{KL}(q(\mathbf{Y}|\mathbf{S})\|p(\mathbf{Y})) - \mathop{\mathbb{E}}_{p(\mathbf{S})} q(\mathbf{Y}|\mathbf{S}),
\end{aligned} \tag{15}$$

which consists of three terms: triple collection probability measure, KL term and regularization of label posterior (positive). We discuss each term in detail.

Recall that the distribution of labeled triples can be represented as follows:

$$s^l \sim \beta\phi_1^l(s^l) + (1-\beta)\phi_0^l(s^l), \tag{16}$$

where ϕ_y^l denotes the probability of being collected over triple space $\mathcal{S}$ for the positive class ($y=1$) and negative class ($y=0$), and $\beta \in [0,1)$ denotes the proportion of true positive samples in labeled data. Similarly, considering the existence of unlabeled positive triples, the distribution of unlabeled samples can be represented as follows:

$$s^u \sim \alpha\phi_1^u(s^u) + (1-\alpha)\phi_0^u(s^u), \tag{17}$$

where $\phi_y^u = 1 - \phi_y^l$ denotes the probability of being uncollected, $\alpha \in [0,1)$ is the class prior or the proportion of positive samples in unlabeled data. Based on that, the first term can be detailed as follows:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{q(\mathbf{Y}|\mathbf{S})}\log p(\mathbf{S}|\mathbf{Y}) = \mathop{\mathbb{E}}_{s^l\in\mathcal{S}^L}\mathop{\mathbb{E}}_{y\in\{0,1\}} q(y|s^l)\log p(s^l|y) \\
&+ \mathop{\mathbb{E}}_{s^u\in\mathcal{S}^U}\mathop{\mathbb{E}}_{y\in\{0,1\}} q(y|s^l)\log p(s^l|y) \\
&= \mathop{\mathbb{E}}_{s^l\in\mathcal{S}^L}\left[q(y=1|s^l)\log[p(s^l|y=1))] + q(y=0|s^l)\log[p(s^l|y=0))]\right] \\
&+ \mathop{\mathbb{E}}_{s^u\in\mathcal{S}^U}\left[q(y=1|s^u)\log[p(s^u|y=1))] + q(y=0|s^u)\log[p(s^u|y=0))]\right] \\
&= \mathop{\mathbb{E}}_{s^l\in\mathcal{S}^L}\left[w^l\log[\phi_1^l(s^l)] + (1-w^l)\log[\phi_0^l(s^l)]\right] \\
&+ \mathop{\mathbb{E}}_{s^u\in\mathcal{S}^U}\left[(w^u\log[\phi_1^u(s^u)] + (1-w^u)\log[\phi_0^u(s^u)]\right],
\end{aligned} \tag{18}$$

where $\mathbf{W}^U = \{w^u\}$ denotes the point-wise probability for the uncollected triples being positive $q(y=1|s^u)$, $\mathbf{W}^L = \{w^l\}$ denotes the probability for the collected triples being positive $q(y=1|s^l)$.

We view $\mathbf{W}^U$ and $\mathbf{W}^L$ as free parameters and regularize them by $\tilde{\mathbf{W}}^U$ and $\tilde{\mathbf{W}}^L$, which are estimated posterior probability as follows:

$$\tilde{w}_i^l = \frac{\beta\phi_1^l(s_i^l)}{\beta\phi_1^l(s_i^l) + (1-\beta)\phi_0^l(s_i^l)}, \tag{19}$$

$$\tilde{w}_{ik}^u = \frac{\alpha\phi_1^u(s_{ik}^u)}{\alpha\phi_1^u(s_{ik}^u) + (1-\alpha)\phi_0^u(s_{ik}^u)}. \tag{20}$$

Therefore, the KL term in Eq. (15) becomes $\mathbb{KL}(\mathbf{W}^U\|\tilde{\mathbf{W}}^U) + \mathbb{KL}(\mathbf{W}^L\|\tilde{\mathbf{W}}^L)$.

Last but not least, the third term $\mathop{\mathbb{E}}_{p(\mathbf{S})} q(\mathbf{Y}|\mathbf{S}) = \|\mathbf{W}^U\|_1/|\mathcal{S}^U| + \|\mathbf{W}^L\|_1/|\mathcal{S}^L|$ regulates the total number of potential triples (including both collected ones and uncollected ones), because of the sparsity nature of graphs. Finally, we derive the lower bound of the log-likelihood, as shown in Eq (13). □

A.2 Datasets

A.2.1 Dataset Information

We evaluate our proposed model based on three widely used knowledge graphs and one Twitter ineraction graph:

Table 3: The statistics of the datasets.

#Ptb Rate	Dataset	#Entities	#Relation	#Train	#Valid	#Test
0.1	**FB15K**	14,951	1,345	340,968	146,129	59,071
	FB15K-237	14,541	237	184,803	79,201	20,466
	WN18	40,943	18	92,428	39,612	5,000
	Twitter	17,839	2	282,233	120,956	110,456
0.3	**FB15K**	14,951	1,345	276,940	118,688	59,071
	FB15K-237	14,541	237	149,229	63,954	20,466
	WN18	40,943	18	72,462	31,055	5,000
	Twitter	17,839	2	232,748	99,749	110,456
0.5	**FB15K**	14,951	1,345	213,380	91,448	59,071
	FB15K-237	14,541	237	113,772	48,759	20,466
	WN18	40,943	18	52,707	22,588	5,000
	Twitter	17,839	2	183,263	78,540	110,456
0.7	**FB15K**	14,951	1,345	150,485	64,493	59,071
	FB15K-237	14,541	237	78,531	33,656	20,466
	WN18	40,943	18	34,984	14,993	5,000
	Twitter	17,839	2	133,778	57,333	110,456

- **FB15K** (Bordes et al., 2013) is a subset of Freebase (Bollacker et al., 2008), a large database containing general knowledge facts with a variety of relation types;

- **FB15K-237** (Toutanova et al., 2015) is a reduced version of FB15K, where inverse relations are removed;

- **WN18** (Bordes et al., 2013) is a subset of WorldNet (Fellbaum, 1998), a massive lexical English database that captures semantic relations between words;

- **Twitter** is an interaction graph relevant with *Russo-Ukrainian War*. Data is collected in the Twitter platform from May 1, 2022, to December 25, 2022, which records user-tweet interactions and user-hashtag interactions. Thus, the graph is formed by two relations (user-tweet and user-hashtag) and multiple entities, which can be categorized into three types (user, tweet, and hashtag). When constructing the graph, thresholds will be selected to remove inactive users, tweets, and hashtags according to their occurrence frequency. We set the thresholds for the user, tweet, and hashtag as 30, 30, and 10, respectively, i.e., if a tweet has fewer than 30 interactions with users, it will be regarded as inactive and removed from the graph. Besides, the extremely frequent users and tweets are deleted as they may be generated by bots.

For all datasets, we first merge the training set and validation set as a whole. Then we simulate noisy and incomplete graphs for the training process by adding various proportions of false negative/positive cases in the merged set. After that, we partition the simulated graphs into new train/valid sets with a ratio of $7:3$ and the test set remains the same. Table 3 provides an overview of the statistics of the simulated datasets corresponding to various perturbation rates and based graphs.

A.2.2 Dataset Perturbation

Data perturbation aims to simulate noisy and incomplete graphs from clean benchmark knowledge graphs. It consists of two aspects: First, to simulate the *false negative issue*, it randomly removes some existing links in a graph, considering the removed links as missing but potentially plausible facts. Second, to simulate the *false positive issue*, it randomly adds spurious links to the graph as unreliable or outdated facts.

We define perturbation rate, i.e., *ptb_rate* , as a proportion of modified edges in a graph to control the amount of removing positive links and adding negative links. For example, if a graph has 100 links and the perturbation rate is 0.5, then we will randomly convert the positivity or negativity of 50 links. Among these 50 modified links, 10% of them will be added and the rest of them will be removed, i.e., we will randomly add 5 negative links and remove 45 positive links to generate a perturbed graph. The perturbed graph can be regarded as noisy and incomplete, leading to significant performance degradation. In our experiments, we set *ptb_rate* in a range of $\{0.1, 0.3, 0.5, 0.7\}$. The detailed perturbation process is summarized in Figure 6.

A.3 Baselines

We describe the baseline models utilized in the experiments in detail:

- **TransE**[2] (Bordes et al., 2013) is a translation-based embedding model, where both entities and relations are represented as vectors in the latent space. The relation is utilized as a translation operation between the subject and the object entity;

- **TransR** (Lin et al., 2015) advances TransE by optimizing modeling of n-n relations, where each entity embedding can be projected to hyperplanes defined by relations;

- **DistMult**[3] (Yang et al., 2014) is a general framework with the bilinear objective for multi-

[2]The experiments of TransE and TransR are implemented with https://github.com/thunlp/OpenKE.

[3]The experiments of DistMult, ComplEx, and RotatE are implemented with https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding.

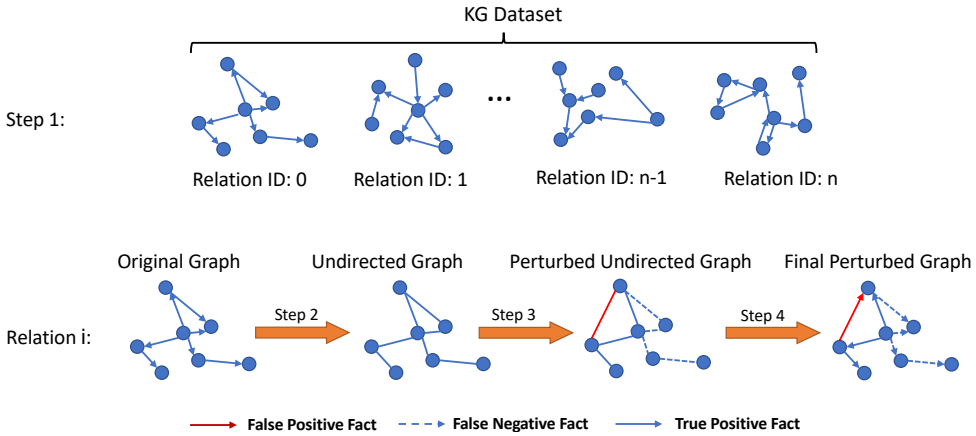


Figure 6: Summary of graph perturbation to construct noisy and incomplete KGs.

relational learning that unifies most multi-relational embedding models;

- **ComplEx** (Trouillon et al., 2017) introduces complex embeddings, which can effectively capture asymmetric relations while retaining the efficiency benefits of the dot product;

- **RotatE** (Sun et al., 2019) extends ComplEx by representing entities as complex vectors and relations as rotation operations in a complex vector space;

- **R-GCN**[4] (Schlichtkrull et al., 2017) uses relation-specific weight matrices that are defined as linear combinations of a set of basis matrices;

- **CompGCN**[5] (Vashishth et al., 2020) is a framework for incorporating multi-relational information in graph convolutional networks to jointly embeds both nodes and relations in a graph;

- **UKGE**[6] (Chen et al., 2019) learns embeddings according to the confidence scores of uncertain relation facts to preserve both structural and uncertainty information of facts in the embedding space;

- **NSCaching**[7] (Zhang et al., 2019) is an inexpensive negative sampling approach by using cache to keep track of high-quality negative triplets, which have high scores and rare;

- **SANS**[8] (Ahrabian et al., 2020) utilizes the rich graph structure by selecting negative samples from a node's k-hop neighborhood for negative sampling without additional parameters and difficult adversarial optimization;

- **PUDA**[9] (Tang et al., 2022) is a KGC method to circumvent the impact of the false negative issue by tailoring positive unlabeled risk estimator and address the data sparsity issue by unifying adversarial training and PU learning under the positive-unlabeled minimax game.

A.4 Reproducibility

A.4.1 Baseline Setup

All baseline models and nPUGraph are trained on the perturbed training set and validated on the perturbed valid set. We utilize *MRR* on the valid set to determine the best models and evaluate them on the clean test set. For uncertain knowledge graph embedding methods UKGE (Chen et al., 2019), since the required uncertainty scores are unavailable, we set the scores for triples in training set as 1 and 0 otherwise. The predicted uncertainty scores produced by UKGE are utilized to rank the potential triples for ranking evaluation. We train all baseline models and nPUGraph on the same GPUs (GeForce RTX 3090) and CPUs (AMD Ryzen Threadripper 3970X 32-Core Processor).

A.4.2 nPUGraph Setup

For model training, we utilize Adam optimizer and set the maximum number of epochs as 200. Within the first 50 epochs, we disable self-training and focus on learning suboptimal model parameters on

[4] https://github.com/JinheonBaek/RGCN
[5] https://github.com/malllabiisc/CompGCN
[6] https://github.com/stasl0217/UKGE
[7] https://github.com/AutoML-Research/NSCaching
[8] https://github.com/kahrabian/SANS
[9] https://github.com/lilv98/PUDA

Table 4: Overall performance on noisy and incomplete Twitter data, with *ptb_rate = 0.3*. Average results on 5 independent runs are reported. $*$ indicates the statistically significant results over baselines, with p-value < 0.01. The best results are in boldface, and the strongest baseline performance is underlined.

Dataset	Twitter			
Metrics	**MRR**	**HIT@100**	**HIT@50**	**HIT@30**
Random	0.001	0.006	0.003	0.002
Knowledge graph embedding methods				
TransE	0.010	0.091	0.058	0.041
TransR	0.009	0.078	0.048	0.033
DistMult	0.021	0.091	0.065	0.052
ComplEx	0.022	0.089	0.064	0.051
RotatE	0.022	0.1115	0.077	0.059
Graph neural network methods on KG				
RGCN	0.005	0.054	0.029	0.019
CompGCN	0.014	0.089	0.059	0.044
Uncertain knowledge graph embedding method				
UKGE	0.011	0.072	0.053	0.033
Negative sampling methods				
NSCaching	0.012	0.095	0.060	0.043
SANS	0.019	0.104	0.070	0.054
Positive-Unlabeled learning method				
PUDA	0.013	0.082	0.057	0.044
nPUGraph	**0.030***	**0.127***	**0.096***	**0.074***
Gains (%)	*38.2*	*13.9*	*25.3*	*26.5*

noisy and incomplete data. After that, we start the self-training strategy, where the latest label posterior estimation is utilized to sample neighbors for the encoder and select informative unlabeled samples for model training. We set batch size as 256, the dimensions of all embeddings as 128, and the dropout rate as 0.5. For the sake of efficiency, we employ 1 neighborhood aggregation layer in the encoder.

For the setting of hyperparameter, we mainly tune positive class prior α in the range of $\{1e-1, 5e-2, 1e-2, 5e-3, 1e-3, 5e-4, 1e-4, 5e-5\}$; true positive ratio β in the range of $\{0.3, 0.2, 0.1, 0.005, 0.001\}$; learning rate in the range of $\{0.02, 0.01, 0.005, 0.001, 0.0005\}$; the number of sampled unlabeled triples for each labeled one in the range of $\{50, 40, 30, 20, 10\}$. We will publicly release our code and data upon acceptance.

A.5 Experiments

A.5.1 Experimental Results on Twitter Data

We discuss the model performance on noisy and incomplete Twitter data with *ptb_rate = 0.3* in this section, which is shown in Table 4. According to the result of Random, we can infer that all relations on Twitter are $n-n$, where relations can be $1-n$, $n-1$, and $n-n$ for benchmark KG. Therefore, link prediction is more challenging for Twitter data and we adopt Hits at $30, 50, 100$ as evaluation metrics, instead.

For Twitter data, nPUGraph achieves impressive performance compared with the baseline models, with 25.98% relative improvement on average. The results of Twitter data support the robustness of nPUGraph, which can mitigate false negative/positive issues not only in benchmark KG but also in the real-world social graph.

A.5.2 Experimental Results under Different Perturbation Rates

The experimental results under perturbation rates 0.1, 0.5, and 0.7 are shown in Table 5, Table 6, and Table 7, respectively. nPUGraph outperforms all baseline models for various perturbation rates, demonstrating that nPUGraph can mitigate false negative/positive issues on knowledge graphs with different degrees of noise and incompleteness. Notably, comparing these three tables, the relative improvements are more significant under higher *ptb_rate* in most cases, showing stronger robustness for nPUGraph on graphs with more false negative/positive facts.

Table 5: Overall performance on noisy and incomplete graphs, with *ptb_rate = 0.1*. Average results on 5 independent runs are reported. $*$ indicates the statistically significant results over baselines, with p-value < 0.01. The best results are in boldface, and the strongest baseline performance is underlined.

Dataset	**FB15K**				**FB15K-237**				**WN18**			
Metrics	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**
Knowledge graph embedding methods												
TransE	0.419	0.666	0.507	0.280	0.245	0.441	0.284	0.144	0.318	0.646	0.579	0.044
TransR	0.398	0.658	0.494	0.250	0.246	0.434	0.280	0.153	0.319	0.646	0.575	0.051
DistMult	0.516	0.719	0.578	0.407	0.271	0.446	0.297	0.185	0.524	0.686	0.610	0.418
ComplEx	0.503	0.711	0.570	0.390	0.276	0.459	0.305	0.186	0.612	0.702	0.643	0.560
RotatE	0.544	0.732	0.608	0.441	0.292	0.480	0.324	0.199	0.613	0.691	0.642	0.567
Graph neural network methods on KG												
RGCN	0.196	0.372	0.209	0.110	0.169	0.317	0.177	0.097	0.483	0.625	0.554	0.396
CompGCN	0.460	0.677	0.525	0.343	0.293	0.475	0.324	0.203	0.608	0.686	0.636	0.564
Uncertain knowledge graph embedding method												
UKGE	0.338	0.425	0.321	0.233	0.231	0.411	0.204	0.110	0.381	0.541	0.407	0.331
Negative sampling method												
NSCaching	0.495	0.689	0.557	0.390	0.153	0.305	0.167	0.080	0.434	0.542	0.470	0.374
SANS	0.422	0.649	0.493	0.298	0.271	0.453	0.301	0.182	0.619	0.702	0.644	0.574
Positive-Unlabeled learning method												
PUDA	0.493	0.713	0.559	0.377	0.271	0.443	0.298	0.185	0.520	0.667	0.608	0.419
nPUGraph	**0.561***	**0.791***	**0.621***	**0.449***	**0.328***	**0.535***	**0.343***	**0.221***	**0.630***	**0.754***	**0.671***	**0.599***
Gains (%)	*3.0*	*8.1*	*2.1*	*1.9*	*12.0*	*11.4*	*5.9*	*8.7*	*1.8*	*7.4*	*4.1*	*4.4*

Table 6: Experimental results under *ptb_rate = 0.5*.

Dataset	**FB15K**				**FB15K-237**				**WN18**			
Metrics	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**
Knowledge graph embedding methods												
TransE	0.279	0.540	0.363	0.136	0.151	0.342	0.190	0.053	0.158	0.337	0.285	0.018
TransR	0.256	0.500	0.323	0.127	0.141	0.294	0.160	0.064	0.150	0.327	0.267	0.020
DistMult	0.328	0.536	0.372	0.224	0.210	0.366	0.228	0.133	0.279	0.370	0.319	0.224
ComplEx	0.322	0.525	0.365	0.220	0.201	0.358	0.220	0.123	0.314	0.373	0.335	0.280
RotatE	0.350	0.547	0.398	0.249	0.227	0.387	0.246	0.149	0.307	0.377	0.333	0.266
Graph neural network methods on KG												
RGCN	0.134	0.271	0.142	0.065	0.116	0.234	0.117	0.058	0.253	0.323	0.287	0.209
CompGCN	0.378	0.600	0.429	0.266	0.223	0.377	0.240	0.149	0.345	0.315	0.336	0.279
Uncertain knowledge graph embedding method												
UKGE	0.257	0.299	0.213	0.088	0.143	0.284	0.172	0.053	0.228	0.297	0.210	0.115
Negative sampling method												
NSCaching	0.272	0.454	0.310	0.179	0.176	0.297	0.190	0.115	0.123	0.182	0.133	0.093
SANS	0.335	0.545	0.387	0.224	0.225	0.382	0.244	0.147	0.313	0.379	0.337	0.275
Positive-Unlabeled learning method												
PUDA	0.329	0.525	0.369	0.229	0.202	0.347	0.217	0.131	0.231	0.329	0.264	0.178
nPUGraph	**0.417***	**0.663***	**0.470***	**0.291***	**0.258***	**0.433***	**0.285***	**0.171***	**0.373***	**0.443***	**0.379***	**0.327***
Gains (%)	*10.4*	*10.5*	*9.6*	*9.6*	*13.9*	*12.0*	*16.1*	*14.8*	*8.2*	*16.9*	*12.6*	*16.9*

Table 7: Experimental results under *ptb_rate = 0.7*.

Dataset	**FB15K**				**FB15K-237**				**WN18**			
Metrics	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**
Knowledge graph embedding methods												
TransE	0.219	0.457	0.294	0.087	0.104	0.273	0.128	0.020	0.114	0.229	0.199	0.020
TransR	0.195	0.396	0.248	0.087	0.096	0.217	0.108	0.036	0.096	0.207	0.167	0.016
DistMult	0.250	0.425	0.282	0.163	0.173	0.308	0.186	0.105	0.181	0.240	0.211	0.143
ComplEx	0.241	0.408	0.271	0.157	0.158	0.290	0.170	0.092	0.202	0.243	0.219	0.178
RotatE	0.271	0.439	0.309	0.185	0.198	0.337	0.212	0.129	0.192	0.250	0.212	0.159
Graph neural network methods on KG												
RGCN	0.129	0.229	0.134	0.075	0.086	0.178	0.086	0.040	0.166	0.215	0.191	0.135
CompGCN	0.354	0.578	0.402	0.243	0.193	0.325	0.204	0.129	0.212	0.262	0.229	0.183
Uncertain knowledge graph embedding method												
UKGE	0.186	0.358	0.199	0.076	0.133	0.201	0.115	0.075	0.099	0.176	0.153	0.116
Negative sampling method												
NSCaching	0.174	0.309	0.194	0.105	0.157	0.276	0.169	0.099	0.057	0.085	0.062	0.041
SANS	0.292	0.478	0.332	0.196	0.201	0.342	0.216	0.131	0.202	0.254	0.222	0.171
Positive-Unlabeled learning method												
PUDA	0.254	0.421	0.283	0.170	0.165	0.291	0.176	0.104	0.109	0.171	0.127	0.077
nPUGraph	**0.365***	**0.600***	**0.427***	**0.277***	**0.243***	**0.390***	**0.266***	**0.155***	**0.247***	**0.303***	**0.257***	**0.209***
Gains (%)	*3.0*	*3.8*	*6.3*	*13.9*	*20.9*	*14.1*	*23.0*	*18.5*	*16.8*	*15.5*	*12.3*	*14.4*