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

Ruijie Wang, Baoyu Li, Yichen Lu, Dachun Sun, Jinning Li, Yuchen Yan, Shengzhong Liu, Hanghang Tong, Tarek F. Abdelzaher
University of Illinois Urbana-Champaign, IL, USA
{ruijiew2, baoyul2, yichen14, dsun18, jinning4, yucheny5, sl29, htong, zaher}@illinois.edu

**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both false negative issue (i.e., potential true facts being excluded) and false positive issue (i.e., unreliable or outdated facts being included).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 nPU-Graph, 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 (Wang et al., 2018b; Saxena et al., 2021; Qian et al., 2019; Wang et al., 2018a, 2022a). 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), and it becomes increasingly vital since it allows for expansion of the existing KG at a low cost. 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 nPU-Graph, 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.

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 unre-
liable 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 \( G = \{(e_h, r, e_t)\} \subseteq S \), where \( S = E \times R \times E \) denotes triple space, \( E \) denotes the entity set, \( R \) denotes the relation set. Each triple \( s = (e_h, r, e_t) \) refers to that
a head entity $e_h \in E$ has a relation $r \in R$ with a tail entity $e_t \in E$. Typically, a score function $\psi(s; \Theta)$, parameterized by $\Theta$, is designed to measure the plausibility of each potential triple $s = (e_h, r, e_t)$, and to rank the most plausible ones to complete KG during inference (Bordes et al., 2013; Sun et al., 2019). The goal of speculative KG reasoning is to infer the most plausible triple for each incomplete triple $(e_h, r, e_t)$ or $(e_t, r, e_t)$ given by sparse and unreliable observations in $G$. In addition, it requires correctness estimation for each potential fact collected or uncollected by $G$.

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 Construction.** For the speculative KG reasoning task, we aim to learn a binary classifier that maps a triple space $S$ to a label space $Y = \{0, 1\}$. Data are split as labeled (collected)\(^1\) triples $s^l \in S^L$ and unlabeled (uncollected) triples $s^u \in S^U$. 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:

$$s^l \sim \beta \phi_L(s^l) + (1 - \beta) \phi_U(s^l),$$

where $\phi_L$ denotes the probability of being collected over triple space $S$ 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:

$$s^u \sim \alpha \phi_U(s^u) + (1 - \alpha) \phi_U^0(s^u),$$

where $\phi_U^0 = 1 - \phi_L^0$ 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 $S^L$ and $S^U$ based on the collected KG $G$. Triples in $G$ naturally serve as labeled samples with a ratio of noise, i.e., $S^L = G$. For unlabeled set $S^U$, however, directly using $S \setminus G$ as the unlabeled set $S^U$ would result in too many unlabeled samples for training due to the large number of possible triples in triple space $S$. Following (Tang et al., 2022), we construct $S^U$ as follows: For each labeled triple $s_k^l = (e_h, r, e_t)$, we construct $K$ unlabeled triples $s^u_{ik}$ by replacing the head and tail respectively with other entities: $s^u_{ik} = (e_h, r, e_k)$ or $(e_k, r, e_t)$, where $e_k$ is the selected entity that ensures $s^u_{ik} \notin S^L$. 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.

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 nPU-Graph 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 \( S^L \) and unlabelled ones \( S^U \), 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 \( \log p(S) \) is lower bounded as follows:

\[
\log p(S) \geq \mathbb{E}_{q(Y)} \left[ \log p(S|Y) - \text{KL}(q(Y)||p(Y)) \right] \\
= \mathbb{E}_{s^L \in S^L} \left[ w^L \log[\phi^L(s^L)] + (1 - w^L) \log[\phi^U(s^L)] \right] \\
+ \mathbb{E}_{s^L \in S^L} \left[ w^U \log[\phi^L(s^U)] + (1 - w^U) \log[\phi^U(s^U)] \right] \\
- \text{KL}(W^L||W^L) - \text{KL}(W^U||W^U) - \frac{||W^L||_2}{|S^L|} - \frac{||W^U||_2}{|S^U|},
\]

(3)

where \( S \) denotes all labeled/unlabeled triples, \( Y \) is the corresponding latent variable indicating the positive/negative labels for triples, \( W^U = \{ w^U_i \} \) denotes the point-wise probability for the uncollected triples being positive, \( W^L = \{ w^L_i \} \) denotes the probability for the collected triples being positive. \( W^U \) and \( W^L \) 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. \( \square \)

We treat label \( Y \) as a latent variable and derive the lower bound for the log-likelihood, which is influenced by the prior knowledge of positive class prior \( \alpha \) and true positive ratio \( \beta \). Thus, maximizing the lower bound can jointly optimize model parameters and infer the posterior label distribution, \( W^U \) and \( W^L \). Such a learning process enables us to avoid false negative/positive issues during model training since it considers \( \phi^L_1 \) (one negative triple is collected) and \( \phi^U_1 \) (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., \( \phi^L_1(\cdot) \) and \( \phi^U_1(\cdot) \) (\( \phi^L_0(\cdot) = 1 - \phi^L_1(\cdot) \)) 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 \( \Theta \), to probability:

\[
\phi^L_1(s) = \sigma(\psi_1(s; \Theta)), \quad \phi^U_1(s) = \sigma(\psi_0(s; \Theta)), \quad (4)
\]

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 \( \phi^U_1(s^U) \) to 0 (\( \phi^U_1(s^U) \) to 1). A loose constraint is that we force the uncollection probability of a collected triple \( s^L \) lower than its corresponding uncollected triples \( s^U \): \( \phi^U_1(s^U) < \phi^U_1(s^L) \). Therefore, we adopt the pair-wise ranking measure \( \phi^U_1(s^L, s^U) \) to replace \( \phi^U_1(s^U) \) as follows:

\[
\phi^U_1(s^U) \rightarrow \phi^U_1(s^L, s^U) = \sigma(\psi_u(s^U; \Theta) - \psi_u(s^L; \Theta)). \quad (5)
\]

**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 \( s^L_i \in S^L \) and its corresponding \( K \) uncollected triples \( s^U_{ik} \in S^U \). We denote the loss function measuring the probability as \( \mathcal{L}_{\text{triple}} \):

\[
\mathcal{L}_{\text{triple}} = -\frac{1}{K|\mathcal{U}|} \sum_{i} \sum_{k} \left[ w^L_i \log[\phi_1^L(s^L_i)] + (1 - w^L_i) \log[\phi_0^L(s^L_i)] \right] \\
+ w^U_i \log[\phi_1^U(s^L_i, s^U_{ik})] \\
+ (1 - w^U_i) \log[\phi_0^U(s^L_i, s^U_{ik})],
\]

(6)
where $w^i_t$ denotes the point-wise probability for the collected triple $s^i_t$ being positive, $w^u_{ik}$ denotes the probability for the uncollected triple $s^u_{ik}$ being positive. Based on the definition, the posterior probability of each collected/uncollected triple being positive can be computed as:

$$w^i_t = \frac{\beta \phi_1(s^i_t)}{\beta \phi_1(s^i_t) + (1 - \beta) \phi_0(s^i_t)}$$ (7)$$

$$w^u_{ik} = \frac{\alpha \phi_1(s^u_{ik})}{\alpha \phi_1(s^u_{ik}) + (1 - \alpha) \phi_0(s^u_{ik})}.$$ (8)

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

$$\min_{\theta} L = \min_{\theta} L_{triple} + L_{KL} + L_{reg},$$ (9)

where $L_{reg} = \|W^L\|_1 + \|W^U\|_1$ can be viewed as a normalization term. Considering the sparsity property of real-world graphs, $L_{reg}$ 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: $h_e, h_r \in \mathbb{R}^d$. To encode more information in $h_e$, 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 $\tilde{W}^L$ 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 $\tilde{W}^L$ and construct the neighbor set $\mathcal{N}_e(\tilde{W}^L) = \{(e_i, r_i)\}$ 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:

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

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

$$\gamma_{e,e_i}^l = \frac{\exp(q_{e,e_i})}{\sum_{(e_j, r_j) \in \mathcal{N}_e(\tilde{W}^L)} \exp(q_{e,e_j})}, \quad q_{e,e_i}^l = a \left( h_e^{l-1} \| h_i^{l-1} \| h_a \right),$$ (11)

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

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

$$\psi_1(s; \Theta) = MLP_1(h_e), \quad \psi_0(s; \Theta) = MLP_0(h_e),$$ (12)

where the MLP input $h_e = [h_{e_h}^l \| h_r \| h_{e_l}^l]$ concatenates entity embeddings and relation embedding.

### 3.4 Self-Training Strategy

The latest label posterior $\tilde{W}^L$ and $\tilde{W}^U$ is further utilized in a self-training strategy to enhance speculative reasoning. First, the latest posterior estimation $\tilde{W}^L$ for collected links updates neighbor sets to gradually prevent the encoder effects by false
positive links. Moreover, the latest estimation \( \tilde{W}^{U} \) for sample unlabeled facts enables us to continuously sample unlabeled triplets with high label posterior to cover positive samples in the unlabeled set to the greatest extent. For each labeled triple \( s_i = (e_h, r, e_t) \), we construct \( K \) unlabeled triplets \( s_{ik}^U \) 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 \( \tilde{\omega}_{ik}^U \). 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 1 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 \( (e_h, r, e^-_t) \) or \( (e^-_t, r, e_t) \), 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 2. 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

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

Table 1: The statistics of the datasets.

---

**Algorithm 1:** Summary of nPUGraph.

| Input: The collected triple set \( \mathcal{S}^L \). |
|---|---|
| Output: The model parameter \( \Theta \), predicted triples. |
| 1 | Construct the unlabeled triple set \( \mathcal{S}^L \) randomly; |
| 2 | Initialize the model parameter \( \Theta \) and the label posterior \( \tilde{W}^L \) and \( W^U \) randomly; |
| 3 | for each training epoch do |
| 4 | Construct neighbor set \( \mathcal{N}_u(\tilde{W}^L) \) by \( \tilde{W}^L \); |
| 5 | Construct unlabeled triple set \( \mathcal{S}^U \) by \( \tilde{W}^U \); |
| 6 | for each collected triple \( s_i \in \mathcal{S}^U \) do |
| 7 | Collect unlabeled triplets \( \{ s_{ik}^U \}_{k=1}^K \); |
| 8 | Calculate \( \phi^U_u(s_i) \) by Eq. (4); |
| 9 | Calculate each \( \phi^U_u(s_i, s_{ik}^U) \) by Eq. (5); |
| 10 | end |
| 11 | Calculate the total loss \( L \) by Eq. (9); |
| 12 | Update model parameter: \( \Theta = \Theta - \frac{\partial L}{\partial \Theta} \); |
| 13 | end |

---
Table 2: 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MRR</th>
<th>HIT@10</th>
<th>HIT@3</th>
<th>HIT@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15K</td>
<td>0.336</td>
<td>0.603</td>
<td>0.425</td>
<td>0.189</td>
</tr>
<tr>
<td>FB15K-237</td>
<td>0.196</td>
<td>0.394</td>
<td>0.236</td>
<td>0.094</td>
</tr>
<tr>
<td>WN18</td>
<td>0.275</td>
<td>0.422</td>
<td>0.275</td>
<td>0.189</td>
</tr>
</tbody>
</table>

Knowledge graph embedding methods

<table>
<thead>
<tr>
<th>Metrics</th>
<th>FB15K</th>
<th>FB15K-237</th>
<th>WN18</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>0.314</td>
<td>0.579</td>
<td>0.397</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.408</td>
<td>0.627</td>
<td>0.463</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.396</td>
<td>0.616</td>
<td>0.451</td>
</tr>
<tr>
<td>RotatE</td>
<td>0.431</td>
<td>0.636</td>
<td>0.489</td>
</tr>
<tr>
<td>RGCN</td>
<td>0.154</td>
<td>0.307</td>
<td>0.164</td>
</tr>
<tr>
<td>CompGCN</td>
<td>0.409</td>
<td>0.631</td>
<td>0.465</td>
</tr>
</tbody>
</table>

Uncertain knowledge graph embedding method

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MRR</th>
<th>HIT@10</th>
<th>HIT@3</th>
<th>HIT@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>UKGE</td>
<td>0.311</td>
<td>0.556</td>
<td>0.337</td>
<td>0.189</td>
</tr>
<tr>
<td>NSCaching</td>
<td>0.371</td>
<td>0.576</td>
<td>0.424</td>
<td>0.265</td>
</tr>
<tr>
<td>SANS</td>
<td>0.372</td>
<td>0.599</td>
<td>0.434</td>
<td>0.252</td>
</tr>
<tr>
<td>PUDA</td>
<td>0.403</td>
<td>0.623</td>
<td>0.458</td>
<td>0.291</td>
</tr>
<tr>
<td>nPUGraph</td>
<td>0.486*</td>
<td>0.718*</td>
<td>0.534*</td>
<td>0.342*</td>
</tr>
</tbody>
</table>

Gains (%)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FB15K</th>
<th>FB15K-237</th>
<th>WN18</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>0.253</td>
<td>0.422</td>
<td>0.275</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.311</td>
<td>0.556</td>
<td>0.337</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.446</td>
<td>0.522</td>
<td>0.471</td>
</tr>
<tr>
<td>RotatE</td>
<td>0.486*</td>
<td>0.718*</td>
<td>0.534*</td>
</tr>
</tbody>
</table>

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.

Table 3: Ablation Studies.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FB15K</th>
<th>FB15K-237</th>
<th>Gains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ablations</td>
<td>MRR</td>
<td>H@10</td>
<td>H@3</td>
</tr>
<tr>
<td>nPUGraph w/o nPU</td>
<td>0.401</td>
<td>0.619</td>
<td>0.293</td>
</tr>
<tr>
<td>nPUGraph w/o LP-Encoder</td>
<td>0.457</td>
<td>0.621</td>
<td>0.261</td>
</tr>
<tr>
<td>nPUGraph w/o Self-Training</td>
<td>0.471</td>
<td>0.704</td>
<td>0.455</td>
</tr>
<tr>
<td>nPUGraph</td>
<td>0.486</td>
<td>0.718</td>
<td>0.534</td>
</tr>
</tbody>
</table>

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 results 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.
We report MRR and Hit@10 over FB15K and FB15K-237 data, as shown in Table 3. 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 number 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 $\alpha$. The positive class prior $\alpha$ and true positive ratio $\beta$ are two important hyperparameters. While $\beta$ has a clear definition from real-world data, the specific value of $\alpha$ is unknown in advance. Figure 5b shows the model performance w.r.t. different values of $\alpha$ by grid search. Reasoning performance fluctuates a bit with different values of $\alpha$ since incorrect prior knowledge of $\alpha$ 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). It has facilitated a wide spectrum of knowledge-intensive applications (Wang et al., 2018b, 2020; Saxena et al., 2021; Qian et al., 2019; Shao et al., 2020; Yang et al., 2020; Yan et al., 2021b,a; Li et al., 2022; Wang et al., 2022a). 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; Wang et al., 2022b, 2023). Recently, graph neural networks (GNN) have been incorporated to enhance representation learning by aggregating neighborhood information (Schlichtkrull et al., 2017; Delmers 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.

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 $\alpha$ and true positive ratio $\beta$. 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.

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.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. Research reported in this paper was sponsored in part by DARPA award HR001121C0165, DARPA award HR00112290105, DoD Basic Research Office award HQ00342110002, the Army Research Laboratory under Cooperative Agreement W911NF-17-20196. It was also supported in part by ACE, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies of DARPA and DoD Basic Research Office or the Army Research Laboratory. The US government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation hereon.

References


A Appendix

A.1 Theorem Proof

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

\[
\log p(S) \geq \mathbb{E}_{q(Y)} \log p(S|Y) - \text{KL}(q(Y)||p(Y))
\]

\[
= \mathbb{E}_{\alpha \in \mathbb{R}^+} \log \left( \frac{p(S|Y)p(Y)}{q(Y)} \right) + \log p(S),
\]

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

\[
\log p(S) \geq \mathbb{E}_{q(Y)} \log \left( \frac{p(S|Y)p(Y)}{q(Y)} \right)
\]

\[
\geq \mathbb{E}_{q(Y)} \log p(S|Y) - \text{KL}(q(Y)||p(Y)) = \mathbb{E}_{q(Y)} q(Y) - \mathbb{E}_{q(Y)} q(Y),
\]

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:

\[
s^l \sim \beta \phi^l_y(s^l) + (1 - \beta) \phi^l_u(s^l),
\]

where $\phi^l_y$ denotes the probability of being collected over triple space $S$ 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:

\[
s^u \sim \alpha \phi^u_y(s^u) + (1 - \alpha) \phi^u_u(s^u),
\]

where $\phi^u_y = 1 - \phi^u_y$ 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:

\[
\mathbb{E}_{\alpha \in \mathbb{R}^+} \log p(S|Y) = \mathbb{E}_{\alpha \in \mathbb{R}^+} \mathbb{E}_{y \in \{0, 1\}} q(y|s^l) \log p(s^l|y) + \mathbb{E}_{\alpha \in \mathbb{R}^+} \mathbb{E}_{y \in \{0, 1\}} q(y|s^u) \log p(s^u|y)
\]

\[
= \mathbb{E}_{\alpha \in \mathbb{R}^+} \mathbb{E}_{y \in \{0, 1\}} q(y|s^l) \log p(s^l|y) + \mathbb{E}_{\alpha \in \mathbb{R}^+} \mathbb{E}_{y \in \{0, 1\}} q(y|s^u) \log p(s^u|y)
\]

\[
= \mathbb{E}_{\alpha \in \mathbb{R}^+} \left[ \mathbb{E}_{\alpha \in \mathbb{R}^+} \left[ q(y=1|s^l) \log[p(s^l|y = 1)] \right] + \mathbb{E}_{\alpha \in \mathbb{R}^+} \left[ q(y=0|s^u) \log[p(s^u|y = 0)] \right] \right]
\]

\[
= \mathbb{E}_{\alpha \in \mathbb{R}^+} \left[ q(y=1|s^l) \log[p(s^l|y = 1)] + q(y=0|s^u) \log[p(s^u|y = 0)] \right]
\]

\[
= \mathbb{E}_{\alpha \in \mathbb{R}^+} \left[ q(y=1|s^l) \log[p(s^l|y = 1)] + q(y=0|s^u) \log[p(s^u|y = 0)] \right]
\]

\[
= \mathbb{E}_{\alpha \in \mathbb{R}^+} \left[ \alpha \phi^u_y(s^u) + (1 - \alpha) \phi^u_u(s^u) \right],
\]

where $\mathbb{W}^U = \{w^u\}$ denotes the point-wise probability for the uncollected triples being positive, $\mathbb{W}^L = \{w^l\}$ denotes the probability for the collected triples being positive.

Proof. Let $\log p(S)$ denote the log-likelihood of all potential triples being collected in the KG or not, $Y$ denote the corresponding latent variable indicating the positive/negative labels for triples, $\mathbb{W}^U = \{w^u\}$ denotes the point-wise probability for the uncollected triples being positive, $\mathbb{W}^L = \{w^l\}$ denotes the probability for the collected triples being positive. We therefore are interested in the difference between the two, measured by the Kullback–Leibler (KL) divergence as follows:

\[
\text{KL}(q(Y)||p(Y)) = \mathbb{E}_{q(Y)} \log \left( \frac{q(Y)}{p(Y)} \right)
\]

\[
= \mathbb{E}_{q(Y)} \log \left( \frac{p(Y)}{q(Y)} \right) + \log p(Y),
\]

\[
\log p(Y) \geq \mathbb{E}_{q(Y)} \log \left( \frac{p(Y)}{q(Y)} \right)
\]

\[
\geq \mathbb{E}_{q(Y)} \log p(Y) - \text{KL}(q(Y)||p(Y)) = \mathbb{E}_{q(Y)} q(Y) - \mathbb{E}_{q(Y)} q(Y),
\]

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

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

\[
\tilde{w}^l_i = \frac{\beta \phi^l_y(s^l)}{\beta \phi^l_y(s^l) + (1 - \beta) \phi^l_u(s^l)},
\]

\[
\tilde{w}^u_i = \frac{\alpha \phi^u_y(s^u)}{\alpha \phi^u_y(s^u) + (1 - \alpha) \phi^u_u(s^u)},
\]

Therefore, the KL term in Eq. (15) becomes $\text{KL}(\mathbb{W}^U||\tilde{\mathbb{W}}^U) + \text{KL}(\mathbb{W}^L||\tilde{\mathbb{W}}^L)$. Last but not least, the third term $\mathbb{E}_{p(Y)} q(Y) = \|\mathbb{W}^U\|_1/|\mathbb{S}^U| + \|\mathbb{W}^L\|_1/|\mathbb{S}^L|$ 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).

\[\square\]

A.2 Datasets

A.2.1 Dataset Information

We evaluate our proposed model based on three widely used knowledge graphs and one Twitter interaction graph.
• **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). Following (Wang et al., 2021, 2022a), 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 1 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-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;

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

\(^3\)The experiments of DistMult, ComplEx, and RotatE are implemented with [https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding](https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding).
• **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 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 \(\alpha\) in the range of \(\{1e^{-1}, 5e^{-2}, 1e^{-2}, 5e^{-3}, 1e^{-3}, 5e^{-4}, 1e^{-4}, 5e^{-5}\}\) and true positive ratio \(\beta\) in the range of \(\{0.3, 0.2, 0.1, 0.005, 0.001\}\); learning rate in the

---

\(^4\)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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metrics</th>
<th>MRR</th>
<th>HIT@100</th>
<th>HIT@50</th>
<th>HIT@30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td></td>
<td>0.001</td>
<td>0.006</td>
<td>0.003</td>
<td>0.002</td>
</tr>
</tbody>
</table>

Knowledge graph embedding methods

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR</th>
<th>HIT@100</th>
<th>HIT@50</th>
<th>HIT@30</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>0.010</td>
<td>0.091</td>
<td>0.058</td>
<td>0.041</td>
</tr>
<tr>
<td>TransR</td>
<td>0.009</td>
<td>0.078</td>
<td>0.048</td>
<td>0.033</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.021</td>
<td>0.091</td>
<td>0.065</td>
<td>0.052</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.022</td>
<td>0.089</td>
<td>0.064</td>
<td>0.051</td>
</tr>
<tr>
<td>RotatE</td>
<td>0.022</td>
<td>0.1115</td>
<td>0.077</td>
<td>0.059</td>
</tr>
</tbody>
</table>

Graph neural network methods on KG

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR</th>
<th>HIT@100</th>
<th>HIT@50</th>
<th>HIT@30</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGCN</td>
<td>0.005</td>
<td>0.054</td>
<td>0.029</td>
<td>0.019</td>
</tr>
<tr>
<td>CompGCN</td>
<td>0.014</td>
<td>0.089</td>
<td>0.059</td>
<td>0.044</td>
</tr>
</tbody>
</table>

Uncertain knowledge graph embedding method

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR</th>
<th>HIT@100</th>
<th>HIT@50</th>
<th>HIT@30</th>
</tr>
</thead>
<tbody>
<tr>
<td>UKGE</td>
<td>0.011</td>
<td>0.072</td>
<td>0.053</td>
<td>0.033</td>
</tr>
</tbody>
</table>

Negative sampling methods

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR</th>
<th>HIT@100</th>
<th>HIT@50</th>
<th>HIT@30</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSCaching</td>
<td>0.012</td>
<td>0.095</td>
<td>0.060</td>
<td>0.043</td>
</tr>
<tr>
<td>SANS</td>
<td>0.019</td>
<td>0.104</td>
<td>0.070</td>
<td>0.054</td>
</tr>
</tbody>
</table>

Positive-Unlabeled learning method

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR</th>
<th>HIT@100</th>
<th>HIT@50</th>
<th>HIT@30</th>
</tr>
</thead>
<tbody>
<tr>
<td>nPUGraph</td>
<td>0.030*</td>
<td>0.127*</td>
<td>0.096*</td>
<td>0.074*</td>
</tr>
</tbody>
</table>

Gains (%) 38.2 13.9 25.3 26.5

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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FB15K</th>
<th>FB15K-237</th>
<th>WN18</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR H@10 H@3 H@1</td>
<td>MRR H@10 H@3 H@1</td>
<td>MRR H@10 H@3 H@1</td>
</tr>
<tr>
<td>TransE</td>
<td>0.419 0.666 0.507 0.280</td>
<td>0.245 0.441 0.284 0.144</td>
<td>0.318 0.646 0.579 0.044</td>
</tr>
<tr>
<td>TransR</td>
<td>0.398 0.658 0.494 0.250</td>
<td>0.246 0.434 0.280 0.153</td>
<td>0.319 0.646 0.575 0.051</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.516 0.719 0.578 0.407</td>
<td>0.271 0.446 0.297 0.185</td>
<td>0.524 0.686 0.610 0.418</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.503 0.711 0.570 0.390</td>
<td>0.276 0.458 0.305 0.186</td>
<td>0.612 0.702 0.643 0.560</td>
</tr>
<tr>
<td>RotatE</td>
<td>0.544 0.723 0.608 0.441</td>
<td>0.292 0.480 0.324 0.199</td>
<td>0.613 0.691 0.642 0.567</td>
</tr>
</tbody>
</table>

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

Table 6: Experimental results under $ptb\_rate = 0.5$.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FB15K</th>
<th>FB15K-237</th>
<th>WN18</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR H@10 H@3 H@1</td>
<td>MRR H@10 H@3 H@1</td>
<td>MRR H@10 H@3 H@1</td>
</tr>
<tr>
<td>TransE</td>
<td>0.279 0.540 0.363 0.136</td>
<td>0.151 0.342 0.190 0.053</td>
<td>0.158 0.337 0.285 0.018</td>
</tr>
<tr>
<td>TransR</td>
<td>0.256 0.500 0.323 0.127</td>
<td>0.141 0.294 0.160 0.064</td>
<td>0.150 0.327 0.267 0.020</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.328 0.536 0.372 0.224</td>
<td>0.210 0.366 0.228 0.133</td>
<td>0.279 0.370 0.319 0.224</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.322 0.525 0.365 0.220</td>
<td>0.201 0.358 0.220 0.123</td>
<td>0.314 0.373 0.335 0.280</td>
</tr>
<tr>
<td>RotatE</td>
<td>0.350 0.547 0.398 0.249</td>
<td>0.227 0.387 0.246 0.149</td>
<td>0.307 0.377 0.333 0.266</td>
</tr>
</tbody>
</table>

Gains (%) 0.561* 0.791* 0.621* 0.449* 0.328* 0.535* 0.343* 0.221* 0.630* 0.754* 0.671* 0.599*

Table 7: Experimental results under $ptb\_rate = 0.7$.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FB15K</th>
<th>FB15K-237</th>
<th>WN18</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR H@10 H@3 H@1</td>
<td>MRR H@10 H@3 H@1</td>
<td>MRR H@10 H@3 H@1</td>
</tr>
<tr>
<td>TransE</td>
<td>0.219 0.457 0.294 0.087</td>
<td>0.104 0.233 0.128 0.020</td>
<td>0.114 0.229 0.199 0.020</td>
</tr>
<tr>
<td>TransR</td>
<td>0.195 0.396 0.248 0.087</td>
<td>0.096 0.217 0.108 0.036</td>
<td>0.096 0.207 0.167 0.016</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.250 0.425 0.282 0.163</td>
<td>0.173 0.308 0.186 0.105</td>
<td>0.181 0.240 0.211 0.143</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.241 0.408 0.271 0.157</td>
<td>0.158 0.290 0.170 0.092</td>
<td>0.202 0.243 0.219 0.178</td>
</tr>
<tr>
<td>RotatE</td>
<td>0.271 0.439 0.309 0.185</td>
<td>0.198 0.337 0.212 0.129</td>
<td>0.192 0.250 0.212 0.159</td>
</tr>
</tbody>
</table>

Gains (%) 0.417* 0.663* 0.470* 0.291* 0.258* 0.433* 0.285* 0.171* 0.373* 0.443* 0.379* 0.327*
ACL 2023 Responsible NLP Checklist

A For every submission:

✔️ A1. Did you describe the limitations of your work?
   Section 7

✔️ A2. Did you discuss any potential risks of your work?
   Section 8

✔️ A3. Do the abstract and introduction summarize the paper’s main claims?
   Abstract and Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
   Left blank.

B ☒ Did you use or create scientific artifacts?

   Left blank.

☐ B1. Did you cite the creators of artifacts you used?
   Not applicable. Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
   Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
   Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
   Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
   Not applicable. Left blank.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
   Not applicable. Left blank.

C ✔️ Did you run computational experiments?

   Section 4

✔️ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
   Section 4.4.2

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4.1, Section A.4

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4, Section A.5

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.