Learning to Extrapolate Knowledge: Transductive Few-shot Out-of-Graph Link Prediction

(2020) - Jinheon Baek, Dong Bok Lee, Sung Ju Hwang

Link:
https://proceedings.neurips.cc/paper/2020/hash/0663a4ddceacb40b095eda264a85f15c-Abstract.html
DOI:
Code/Link:
Tags: #paper
Rating:

Abstract

Many practical graph problems, such as knowledge graph construction and drug-drug interaction prediction, require to handle multi-relational graphs. However, handling real-world multi-relational graphs with Graph Neural Networks (GNNs) is often challenging due to their evolving nature, as new entities (nodes) can emerge over time. Moreover, newly emerged entities often have few links, which makes the learning even more difficult. Motivated by this challenge, we introduce a realistic problem of few-shot out-of-graph link prediction, where we not only predict the links between the seen and unseen nodes as in a conventional out-of-knowledge link prediction task but also between the unseen nodes, with only few edges per node. We tackle this problem with a novel transductive meta-learning framework which we refer to as Graph Extrapolation Networks (GEN). GEN meta-learns both the node embedding network for inductive inference (seen-to-unseen) and the link prediction network for transductive inference (unseen-to-unseen). For transductive link prediction, we further propose a stochastic embedding layer to model uncertainty in the link prediction between unseen entities. We validate our model on multiple benchmark datasets for knowledge graph completion and drug-drug interaction prediction. The results show that our model significantly outperforms relevant baselines for out-of-graph link prediction tasks.

Introduction

1. [GNN]: Exploit the graph-structured data which works on a non-Euclidean domain. Mostly deals with simple graphs with unlabeled edges.

2. [Relation-aware GNNs]: Considers mluti-relational graphs with labels and directions on the edges.
    - Natural language understanding
    - Modeling protein structure
    - Drug-drug interaction prediction
    - Retrosynthesis planning

Challenges of link prediction for KGs

1. KGs dynamically evolve over time. Around 200 new entities emerge every day. Predicting links on the emerging (Unseen) entities pose a new challenge.
2. Real-world KGs exhibit [long-tail distributions], where large portion of the entities have only few triplets. [Embedding-based methods] cannot embed unseen entities.

Propose Graph Extrapolation Networks (GENs)

- Few-shot Out-of-Graph(OOG) link prediction for emerging entities
- Seen - Unseen && Unseen - Unseen
- GENs are meta-learned to extrapolate the knowledge from seen to uness entities
- Transfers knowledge from entities with many to few links.
- Meta-trains two GNNs to predict the links between entities
    1. [Inductive GEN]: 1st GNN; learns to embed the unseen entities that are not observed, and predicts the links between seen and unseen entities.
    2. [Transductive GEN]: 2nd GNN; learns to predict the links not only between seen and unseen entities, but also between unseen entities themselves.
- Link prediction for unseen entities is unreliable, which gets worse when few triplets are available for each entitiy.
- [Stochastic embedding] is used to learn the distribution of unseen representations for stochastic embeeding to account for the [uncertainty]
- [Transfer learning strategy] to model the long-tail distribution. Leads GEN to represent the unseen entities that are well aligned with the seen entities

Link prediction goal

- Entitiy prediction
- Relation prediction

Learning to Extrapolate Knowledge with Graph Extrapolation Networks

- Training for 'seen-to-seen' work well on 'seen-to-unseen' cases?

- No. Use [meta learning] framdework to handle the OOG link prediction problem. Train a model over a distribution of tasks such that the model generalizes well on unseen tasks.

Learning Objective

- Maximize the score of a true triplet $s(e_h, r, e_t)$ that contains any unseen entities e' to rank it higher than all the other false triplets, with embedding and score function parameters θ denoted as follows:

$$\max_{\theta} \mathbb{E}_{e' \sim p(\mathcal{E}')} \left[s(e', r, \tilde{e}; \theta) \text{ or } s(\tilde{e}, r, e'; \theta) \right], \quad \text{where} \quad \tilde{e} \in (\mathcal{E} \cup \mathcal{E}') \text{ and } e' \in \mathcal{E}'.$$

- Generalization to real unseen entities can be tackled with [meta-learning] by simulating unseen entities during training

Meta-Learning Framework

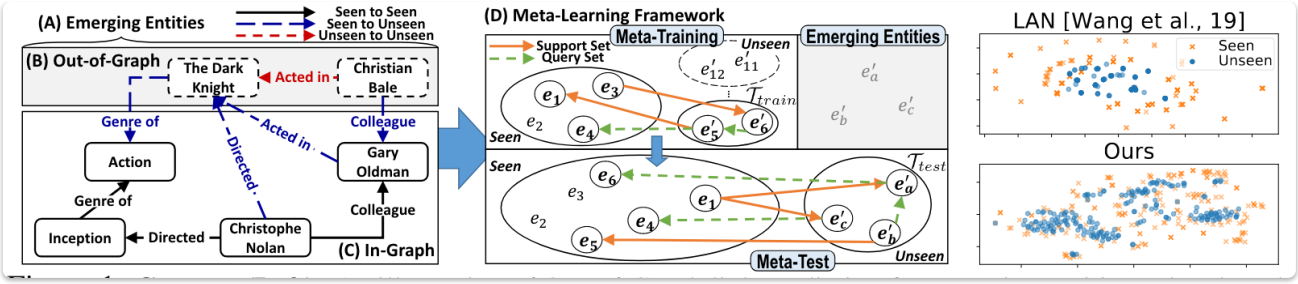
- We can formulate a set of tasks such that the model learns to generalize over unseen entities, which are simulated using seen entities.

1. Randomly split the entities in a given graph into the [meta-training set] for simulated unseen entities, and the [meta-test set] for real unseen entities
2. Generate a task by sampling the set of simulated unseen entities during meta-training, for the learned model to generalize over actual unseen entities.
3. Each task $\mathcal{T}$ over a distribution $p(\mathcal{T})$ corresponds to a set of unseen entities $\mathcal{E}_{\mathcal{T}} \subset \mathcal{E}'$ with a predefined number of instances $|\mathcal{E}_{\mathcal{T}}| = N$
4. Divide the triplets associative with each entity $e'_i \in \mathcal{E}_{\mathcal{T}}$ into the [support set] S_i and the [query set] $\mathcal{Q}_i : \mathcal{T} = \bigcup_{i=1}^{N} S_i \cup \mathcal{Q}_i$

- Meta-objective: Learning to represent the [unseen entities] as ϕ using a [support set] S with a [meta-function] f, to maximize the triplet score on a [query set] $\mathcal{Q}$ with a [score function] s as follows:

$$\max_{\theta} \mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})} \left[\frac{1}{N} \sum_{i=1}^{N} \frac{1}{|\mathcal{Q}_i|} \sum_{j=K+1}^{M_i} s(e'_i, r_j, \tilde{e}_j; \phi_i, \theta) \text{ or } s(\tilde{e}_j, r_j, e'_i; \phi_i, \theta) \right], \ \phi_i = f_\theta(\mathcal{S}_i).$$

- Once the model is trained with the [meta-training tasks] $\mathcal{T}_{train}$, we can apply it to [unseen meta-test tasks] $\mathcal{T}_{test}$. The set of entities is disjoint from $\mathcal{T}_{train}$.



Inductive Graph Extrapolation Networks (I-GEN)

- GNN-based meta-learner that ouputs the representation of unseen entities.
- Extrapolates knowledge of a given graph $\mathcal{G}$ to an unseen entity e_i' through a [support set] S_i. We formulate $f_\theta(\cdot)$ as follows.

$$f_\theta(S_i) = \frac{1}{K} \sum_{(r,e) \in n(S_i)} W_r C_{r,e}$$

- $n(\cdot)$ is a set of neighboring entities and relations
- $n(S_i) = (r, e) | (e_i', r, e) \text{ or } (e, r, e_i') \in S_i$
- K is a size of $n(S_i)$
- $W_r \in \mathbb{R}^{d \times 2d}$ is a relation-specific transformation matrix that is meta-learned
- $C_{r,e} \in \mathbb{R}^{2d}$ is a concatenation of feature representation of the relation-entity pair.

Transductive Meta-Learning of GENs (T-GEN)

- Drawback of [Inductive GEN]: Does not consider the relationships between unseen entities.
- We add one more GEN layer $g_\theta(\cdot)$ to consider inter-relationships between unseen entities.

$$g_\theta(S_i, \phi) = \frac{1}{K} \sum_{(r,e) \in n(S_i)} W_r' C_{r,e} + W_0 \phi_i$$

- $W_0 \in \mathbb{R}^{d \times d}$ is a weight matrix for the self-connection to consider the embedding ϕ_i which is updated by the [previous inductive layer] $f_\theta(S_i)$
- Transductive layer $g_\theta(\cdot)$ aggregates the representation across all the neighbors with a weight matrix $W_r' \in \mathbb{R}^{d \times 2d}$ where neighbors can include the unseen entities with embeddings ϕ, rather than treating them as noises or ignoring them as zero vectors like [inductive scheme].

Stochastic Inference

- A [naive transductive GEN] generalizes to the unseen entities by simulating them with the seen entities during meta-training.
- However, due to the intrinsic unreliability of [OOG] with each entity having only few triplets, there could be high uncertainties on the representations of unseen entities.
- To model uncertainties, stochastically embed the unseen entities by learning the distribution over an unseen entity embedding ϕ_i' .
- Approximate the posterior using $q(\phi_i'|S_i, \phi) = N(\phi_i'|\mu_i, \text{diag}(\sigma_i^2))$
- Then compute the mean and variance via [two individual transductive GEN layers] $\mu_i = g_{\theta_\mu}(S_i, \phi)$ and $\sigma_i = g_{\theta_\sigma}(S_i, \phi)$, which modifies the GraphVAE to our setting
- Maximize the score function s

$$s(e_h, r, e_t) = \frac{1}{L}\sum_{l=1}^{L} s(e_h, r, e_t; \phi^{'(l)}, \theta), \quad \phi^{'(l)} \sim q(\phi'|S, \phi)$$

- Set MC samples size to $L = 1$ during [meta-training] for computational efficiency
- Set MC samples size to $L = 10$ during [meta-testing] for MC approximation
- Let the approximate posterior same as the prior to make the consistent pipeline at training and test
- Model the source of uncertainty on the output embedding of an unseen entity from the [Transductive GEN] layer via Monte Carlo dropout
- Final GEN is trained for both the inductive and transductive steps with stochastic inference

Loss function

- Each task $\mathcal{T}$ that corresponds to a set of unseen entities $\mathcal{E}_\mathcal{T} \subset \mathcal{E}'$ consists of a [support set] and [query set]: $\mathcal{T} = \mathcal{S}, \mathcal{Q}$
- During [training], represent the embeddings of unseen entities $e_i' \in \mathcal{E}_\mathcal{T}$ using the support set S with [GENs]
- During [testing], use the [true labeled query set] $\mathcal{Q}_i$ to optimize [GENs]
- Every query set contains only [positive triplets], perform [negative sampling] to update a meta-learner by allowing it to distinguish positive from negative triplets.
- Specifically, replace the entity of each triplet in the [query set] $\mathcal{Q}_i^- = (e_i', r, e^-) \text{ or } (e^-, r, e_i') \mid e^- \in \mathcal{E}$ where e^- is the corrupted entity.
- Now, $\mathcal{Q}_i^-$ holds negative samples for an unseen entity e_i'

- Final loss is below

$$\mathcal{L}\left(\mathcal{Q}_i\right)=\sum_{\left(e_h, r, e_t\right) \in \mathcal{Q}_i} \sum_{\left(e_h, r, e_t\right)^{-} \in \mathcal{Q}_i^{-}} \max \left\{\gamma-s^{+}\left(e_h, r, e_t\right)+s^{-}\left(e_h, r, e_t\right)^{-}, 0\right\},$$

Meta-Learning for Long-Tail Tasks

- Real-world graphs follow the long-tail distributions
- It's benefitial to transfer the knowledge from entities with many links to entities with few links
- Start to learn the model with many shot cases, then gradually decrease the number of shots to few shot cases in a logarithmic scale