Pattern Recognition Letters
journal homepage: www.elsevier.com

Locality-aware subgraphs for inductive link prediction in knowledge graphs

Hebatallah A. Mohamed[a,**], Diego Pilutti[a], Stuart James[b], Alessio Del Bue[b], Marcello Pelillo[a,c], Sebastiano Vascon[a,c,**]

[a]*European Center for Living Technology (ECLT), Ca' Foscari University of Venice, Italy*
[b]*Pattern Analysis and Computer Vision (PAVIS), Istituto Italiano di Tecnologia (IIT), Italy*
[c]*Department of Environmental Sciences, Informatics and Statistics (DAIS), Ca' Foscari University of Venice, Italy*

Article history:

Knowledge graphs, Inductive link prediction, Graph neural networks, Local clustering, Personalized PageRank

ABSTRACT

Recent methods for inductive reasoning on Knowledge Graphs (KGs) transform the link prediction problem into a graph classification task. They first extract a subgraph around each target link based on the k-hop neighborhood of the *target entities*, encode the subgraphs using a Graph Neural Network (GNN), then learn a function that maps subgraph structural patterns to link existence. Although these methods have witnessed great successes, increasing k often leads to an exponential expansion of the neighborhood, thereby degrading the GNN expressivity, due to oversmoothing. In this paper, we formulate the subgraph extraction as a local clustering procedure that aims at sampling tightly-related subgraphs around the target links, based on a Personalized PageRank (PPR) approach. Empirically, on three real-world KGs, we show that reasoning over subgraphs extracted by PPR-based local clustering can lead to a more accurate link prediction model than relying on neighbors within fixed hop distances. Furthermore, we investigate graph properties such as average clustering coefficient and node degree, and show that there is a relation between these and the performance of subgraph-based link prediction.

1. Introduction

A Knowledge Graph (KG) is a large directed network of real-world entities and relationships between them, where facts are represented as triplets in the form of (*head entity*, *relation*, *tail entity*). KGs have brought important solutions to many real-world applications, such as semantic parsing [1, 2], information extraction [3, 4], object detection [5], scene graph generation [6], and question answering [7, 8]. Despite the huge amounts of relational data, one of the major challenges is that KGs typically suffer from incompleteness, as links between the entities are often missing. Therefore, link prediction has become a fundamental task [9], that aims at estimating the likelihood of the existence of relations between KG entities.

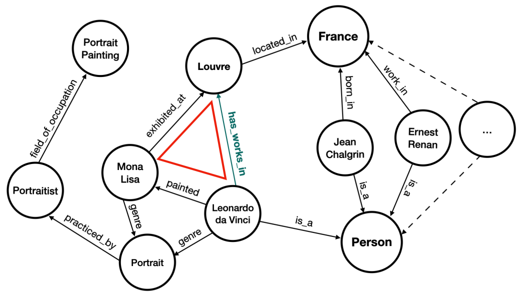


Figure 1: An example of a Knowledge Graph: (*France* and *Person* are hub nodes – connected to many other nodes in the graph). A logical rule that can be extracted (highlighted with a red triangle): if (X, *painted*, Y) and (Y, *exhibited_at*, Z), then (X, *has_works_in*, Z).

Numerous embedding-based methods have been developed for link prediction, by representing entities and relations in a low-dimensional space [10, 11, 12, 13]. However, those methods are limited to the transductive setting, where every entity at inference time must be seen during training. The inductive

**Corresponding authors
e-mail: hebatallah.mohamed@unive.it (Hebatallah A. Mohamed), diego.pilutti@unive.it (Diego Pilutti), stuart.james@iit.it (Stuart James), alessio.delbue@iit.it (Alessio Del Bue), marcello.pelillo@unive.it (Marcello Pelillo), sebastiano.vascon@unive.it (Sebastiano Vascon)

link prediction task, which is the focus of this paper, is more challenging since it aims at predicting missing links between entities, where entities during training and inference can be different [14, 15, 16]. Inductive entity-agnostic methods predict missing links by learning logical rules from the KG. For example, from the KG shown in Figure 1, a rule that can be induced: If (X, *painted*, Y) and (Y, *exhibited_at*, Z), then (X, *has_works_in*, Z).

The common strategy of entity-agnostic reasoning is to (i) extract an enclosing subgraph around each target link, (ii) encode the subgraphs using a Graph Neural Network (GNN) [17], and (iii) learn a function that maps subgraph structural patterns to a link's presence. Existing methods extract the subgraphs by incorporating all nodes indistinguishably at a given number of hops from the target links. In such methods, subgraph node importance is computed by a fixed scheme based on its distance with the target link, then added as node feature [15, 14]. SEAL [14] has proved that a relatively small number of hops, such as 3-hop, is enough for learning high-order heuristics. However, even 3-hop frequently results in very large subgraphs, especially if nodes with high degree (hubs) are included [14, 18].

GNNs have some limitations when integrating node features and capturing topological patterns in large and complex graphs due to oversmoothing [19, 20, 21, 22]. Therefore, increasing the size of the neighborhood around the target entities does not improve (or may decrease) the GNN predictive performance. In this paper, we seek to address whether reasoning on locality-aware subgraphs around the target link can produce higher link prediction accuracy than reasoning over subgraphs extracted by a fixed number of hops. Thus, our research questions are: *(RQ1) does the locality of training (and inference) data have an effect on the accuracy of link prediction* and, if so, *(RQ2) how much local data can effectively train a link prediction model?*.

Inspired by previous studies [23, 24, 25, 26], we utilize the strong localization properties of the personalized PageRank (PPR) [27] to preserve more meaningful neighborhood (i.e., context) around a target link in the graph. More precisely, we use PPR to perform locally-biased random walks around given seed nodes (the *head* and *tail* of a target link), which allows us to rank the importance of nodes in a graph from a target link. We then rely on PPR scores to extract local and dense subgraphs near the target links, for the training and inference phases of a proposed GNN-based link prediction model.

In brief, the main contributions of this paper are as follows:

- Introducing a novel strategy, named LCILP (Local Clustering for Inductive Link Prediction), for inductive link prediction, by reasoning over locally-aware subgraphs extracted by PPR-based local clustering technique.
- Studying the relation between graph properties and our proposed approach. We show that that there is a relation between the average clustering coefficient, the average node degree of a graph, and the performance of link prediction.
- Evaluating our approach on three benchmark datasets for inductive link prediction. Our model demonstrates a significant gain in performance compared to state-of-the-art models in terms of AUC-PR and Hits@10.

2. Related work

2.1. Embedding-based link prediction

Various methods that represent KG entities and relations in a low-dimensional space have been proposed, such as TransE [10], TransH [11], ComplEx [12] and RotatE [13]. Those methods have focused on transductive link prediction; they rely on the fact that entities at inference time were seen during training. To cope with this limitation, inductive representation learning approaches have been proposed. However, inductive approaches do not necessarily focus on exploiting local structural patterns in KG, which is strictly related to inductive reasoning. For instance, some approaches have leveraged node features [28, 29], but there are many KGs without node features. Other works, such as [30], need the unseen nodes to be surrounded by seen nodes and can not handle entirely new graphs.

2.2. Rule induction-based link prediction

Several rule learners, including Neural-LP [31], RuleN [32], and DRUM [33], have been proposed to learn entity-independent logical rules in KGs for link prediction. However, such models suffer from the problem of scaling to large datasets [15]. Therefore, deep learning methods have been developed. These methods mainly focus on learning heuristics from enclosing subgraphs around the target links using a GNN [17]. For example, SEAL [14] induces enclosing subgraphs based on the union of target nodes' neighbors up to k-hop, then encodes the structural patterns of the subgraphs using a GNN [34]. GraIL [15] extends SEAL to support directed multi-relational KGs, by replacing the classical GNN with multi-relation R-GCN [35]. In GraIL, the enclosing subgraph is induced by all the nodes that occur on a path between the two target nodes; it is given by the intersection of the k-hop neighborhood nodes of those two target nodes. TACT [16] extends GraIL by modeling relation correlation patterns. However, in TACT, the authors report results of relation classification; predicting *relation* between a given (*head*, *tail*), while in SEAL and GraIL, the authors evaluate their models on link prediction by inferring *head*, given (*relation*, *tail*) and vice versa. Other hybrid approaches that incorporate prior logic rules into the GNN are proposed [36, 37, 38]. Although these methods have achieved good performance, they lack effective relation modeling and cannot be applied to inductive settings.

2.3. Graph sampling for GNNs

There are some works that have focused on sampling subgraphs to design more efficient and scalable algorithms for training deep and large GNNs. For example, GraphSAGE [28] proposes the idea of neighborhood sampling, by first down-sampling node neighborhoods randomly to a fixed-size set of nodes, then aggregating the sampled ones. GraphSAINT [39]

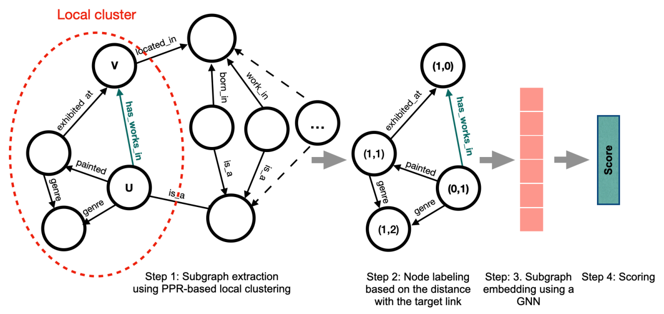


Figure 2: Our approach computes a local sub-graph selected by PPR around the target set and applies node labeling as input to a GNN to score the relation.

proposes random walk samplers to construct mini-batches during GNN training. Cluster-GCN [40] adopts a global graph partitioning algorithm to partition the input graph into subgraphs, and run a GNN on each subgraph. Alternatively, DropEdge [41] randomly removes a certain number of edges from the input graph at each training epoch. Recently, PPNP [42] and PPRGo [24] propose to use PPR for scaling GNN for semi-supervised node classification. However, those works have focused on node classification; less attention has been given to link prediction. In this paper, we aim to enhance the inductive link prediction, by reasoning over the most relevant parts of the graph, using a PPR-based approach.

3. Approach

To perform inductive link prediction, our model aims to score the plausibility of a target triplet (u, r_t, v), where r_t is a target relation between a *head* node u and *tail* node v in a KG. We propose an approach that consists of four main sequential steps: (1) extracting a subgraph around u and v using a PPR-based local clustering technique, (2) labeling the extracted subgraph nodes, (3) encoding the subgraph using a GNN model, (4) scoring the subgraph. Figure 2 gives an overview of the approach. We detail each of the four steps in the following subsections.

3.1. Step 1: Subgraph extraction

The subgraph extraction consists of two steps: first, nodes are scored according to their proximity to a given *seed set*, representing the *target entities*. Secondly, nodes are considered in decreasing order of their score to create nested local clusters, which can then be evaluated using a goodness metric.

PPR, also known as *random-walk-with-restart*, is one of the most common scoring methods that measure node importance [43, 44, 45]. We use an approximate PPR to overcome the problem of PPR being computationally expensive [43]. As shown in Algorithm 1, we rank the vertices of a graph G based on the *seed set* $\Omega = \{u, v\}$. The approximation parameter ϵ maintains two vectors: the solution vector p and a residual vector r, where p vector is the approximation of the PPR vector and vector r contains the approximation error. The teleportation probability α controls the amount of information we are incorporating from the neighborhood of the *seed set*. Namely, when the value of α is close to 1, the random walks teleport to the seed nodes more often, and we are therefore placing more

Algorithm 1 Approximate PPR-based ranking

Input: Graph $G = (V, E)$ with vertices V and edges E, seed set $\Omega = \{u, v\}$, teleportation probability $\alpha \in (0, 1]$, residual error ϵ.
Output: A sequence of nested sets of vertices ranked by the approximate PPR.

```
 1: Initialize PageRank vector $p(x) = 0 \quad \forall x \in V$;
 2: Initialize $r(x) = \begin{cases} \frac{1}{2} & \text{if } x \in \Omega, \\ 0 & \text{otherwise.} \end{cases} \quad \forall x \in V$;
 3: while $\max_{x \in V} \frac{r(x)}{\deg(x)} \geq \epsilon$, where deg(x) is the degree of node $x$ do
 4:     $\acute{r} = r$;
 5:     $p(x) = p(x) + \alpha r(x)$;
 6:     $\acute{r}(x) = (1 - \alpha) r(x)/2$;
 7:     for each $y$ such that $(x, y) \in E$ do
 8:         $\acute{r}(y) = r(y) + (1 - \alpha) r(x)/(2\deg(x))$;
 9:     end for
10:     $r = \acute{r}$;
11: end while
12: Sort vertices in descending order of their score, so that $p(x_1) \geq p(x_2) \geq ... \geq p(x_J)$, where $J$ is the number of vertices with non-zero scores;
13: return $S = \{S_1 \subset S_2 \subset, ..., \subset S_j\}$, where $S_j = \{x_i | i \leq j\}$, and $j \in \{1, ..., J\}$.
```

importance on the immediate neighborhood of the nodes. As the value of α decreases, we instead give more importance to the extended (multi-hop) neighborhood of the seed nodes.

Algorithm 1 produces a sequence S of nested sets of vertices representing local clusters, $S = \{S_1 \subset S_2 \subset, ..., \subset S_j\}$, ranked by the PPR. Given a j-th set in S, we measure its quality using the *conductance*, which determines how tight-knit a set of vertices is in a graph [43]. The conductance of $S_j \subseteq V$ is calculated as:

$$conductance(S_j) = \frac{\text{cut}(S_j)}{\min\left(\text{vol}(S_j), \text{vol}(\hat{S}_j)\right)}, \tag{1}$$

where $\text{cut}(S_j)$ is the number of edges with one end point in S_j and the other end point in the complement set $\hat{S}_j = V \setminus S_j$, and $\text{vol}(S_j)$ is the number of edge end points in S_j. The conductance of a set S_{j+1} can be computed from the conductance of S_j. Among the sets S, we select the set S_j with the lowest conductance since it indicates a good local cluster; cluster vertices are densely connected with the target link, and sparsely connected with the rest of the graph. We then prune the nodes that do not occur on a path between u and v. The final outcome from this process is a subgraph reflecting the local cluster to be used for reasoning, defined as $G(u, r_t, v)$. The running time of calculating the PageRank vector p is $O\left(\frac{1}{\epsilon\alpha}\right)$, while the complexity of sorting p and calculating the conductance is $O\left(|\text{Supp}(p)| \log(|\text{Supp}(p)|) + \text{vol}(\text{Supp}(p))\right)$, where Supp(p) is the set of non-zero vertices.

3.2. Step 2: Node labeling

Afterward, we define an entity-independent embedding for each entity (node) in the subgraph. Following [15], each node i in the subgraph around u and v nodes of the target relation is labeled with the tuple $(d(i,u), d(i,v))$, where $d(i,u)$ is the shortest distance between nodes i and u (likewise for $d(i,v)$). The two target nodes, u and v, are labeled (0, 1), and (1, 0) to be identifiable by the model. This scheme captures the position of each node in the subgraph with respect to the target link. The node features are defined as [one-hot$(d(i,u)) \oplus$ one-hot$(d(i,v))$], representing the concatenation of the one-hot embedding of the labels. Therefore, the dimension of node features is bounded by the maximum number of hops in the extracted subgraphs.

3.3. Step 3: Subgraph embedding

We then use a multi-relational R-GCN [35], but any relational GNN could be used, to learn the embeddings of the extracted subgraph $G(u, r_t, v)$. R-GCN adopts a general message-passing scheme [46], where a node representation is iteratively updated by combining it with the aggregation of its neighbors' representation.

In the k-th layer of our GNN, a_i^k represents the aggregated message from the neighbors of node i. The aggregation function is defined as:

$$a_i^k = \sum_{r=1}^{R} \sum_{s \in N_r(i)} \alpha^k_{r(s,i)} W_r^k h_s^{k-1}, \tag{2}$$

where R is the total number of unique relation types , $N_r(i)$ represents the neighbors of node i under relation r, W_r^k is the transformation matrix of the k-th layer over relation r, and $\alpha_{r(s,i)}$ is the edge attention weight at the k-th layer corresponding to the edge between nodes s and i via relation r.

The latent representation of node i in the k-th layer is:

$$h_i^k = \mathrm{ReLU}(W_0^k h_i^{k-1} + a_i^k), \tag{3}$$

where W_0 is a transformation matrix that aims at retaining the information of the node itself using a self-loop, and ReLU is an activation function.

The subgraph representation of $G(u, r_t, v)$ is obtained by average-pooling of all the node representations:

$$h^L_{G(u,r_t,v)} = \frac{1}{|V|} \sum_{i \in V} h_i^L, \tag{4}$$

where V denotes the set of vertices in $G(u, r_t, v)$, and L represents the number of layers of message-passing.

3.4. Step 4: Scoring and loss function

For scoring the likelihood of a triplet (u, r_t, v), we concatenate the subgraph representation $h_G^L(u, r_t, v)$, the target nodes' latent representations (h_u^L and h_v^L), and a learned embedding of the target relation (e_{r_t}), then pass these concatenated representations through a linear layer:

$$score(u, r_t, v) = [h_G^L(u, r_t, v) \oplus h_u^L \oplus h_v^L \oplus e_{r_t}]W, \tag{5}$$

where $\oplus$ refers to the concatenation operation, and W is a learnable weight matrix.

Finally, we train the model to score positive triplets higher than negative ones. More precisely, for each triplet in the training KG, we sample a negative triplet by replacing the head or tail of the triplet with a uniformly sampled random entity. We then use the following loss function to train our model via stochastic gradient descent:

$$L = \sum_{i=1}^{|T|} \max\left(0, score(u_i, r_{t_i}, v_i) - score(\acute{u_i}, \acute{r_{t_i}}, \acute{v_i}) + \gamma\right), \tag{6}$$

where T is the set of triplets in the training graph, (u_i, r_{t_i}, v_i) and $(\acute{u_i}, \acute{r_{t_i}}, \acute{v_i})$ are the positive and negative triplets respectively, and γ is the margin hyperparameter.

4. Experiments

4.1. Datasets

We perform our experiments considering three state-of-the-art publicly available benchmark datasets for inductive link prediction proposed in GraIL [15]: (1) WN18RR [47]: is derived from WordNet [48], in which entities correspond to word senses, and relationships define lexical relations between them; (2) FB15K-237 [49]: is a subset of Freebase [50], a large KG of general facts, mostly about movies, actors, awards and sports; (3) NELL-995 [51]: is a KG constructed from high confidence facts of NELL system [52].

Each of the datasets has four versions with different numbers of relations, number of links, and connectivity properties. Being an inductive task, the datasets were sampled by ensuring that train and test sets do not have overlapping entities. In Table 1, we summarize the statistics and structural properties of the datasets. We report the average degree of the nodes since the Approximate PPR algorithm is highly dependent on node degrees. Furthermore, we report the average clustering coefficient, which is a measure of the degree to which nodes in a graph tend to cluster together [53]. The average clustering coefficient is computed as follows:

$$AvgC = \frac{1}{n} \sum_{i=1}^{n} C(i), \tag{7}$$

where n is the number of nodes in the graph, and $C(i)$ is the local clustering coefficient for given node i given by:

$$C(i) = \frac{2\mathrm{T}(i)}{\deg(i)(\deg(i) - 1)}, \tag{8}$$

where $\mathrm{T}(i)$ is the number of triangles through node i, and $\deg(i)$ is the degree of node i. The average clustering coefficient over both the train and test is computed as follows:

$$AvgC\text{-}2 = \tfrac{1}{n_{tr}+n_{te}} \left(\textstyle\sum_{i=1}^{n_{tr}} C(i) + \sum_{i=1}^{n_{te}} C(i) \right). \tag{9}$$

where n_{tr} and n_{te} is the number of nodes in the training and testing graphs, respectively. Topology-based heuristics often rely on triangular patterns to extract logical rules [54, 55]. Hence, we assume that graphs with a high average clustering coefficient will have a high performance of inductive link prediction, in comparison to the ones with a low clustering coefficient. It is worth noting that we do not use the *AvgC* or *AvgC-2* during training or testing our model, but only to assess the relationship between the performance of our model and the underlying structure of the KGs.

Table 1: Statistics of the inductive benchmark datasets: number of unique relations (*#Rel*), number of nodes (*#Node*), number of links (*#Link*), average node degree (*Deg*), average clustering coefficient per train/test set (*AvgC*), and average clustering coefficient considering the nodes in both train and test sets (*AvgC-2*).

		WN18RR						FB15k-237						NELL-995					
		#Rel	*#Node*	*#Link*	*Deg*	*AvgC*	*AvgC-2*	*#Rel*	*#Node*	*#Link*	*Deg*	*AvgC*	*AvgC-2*	*#Rel*	*#Node*	*#Link*	*Deg*	*AvgC*	*AvgC-2*
v1	train	9	2746	6678	2.43	0.087	0.091	183	2000	5226	2.61	0.115	0.097	14	3103	5540	1.79	0.088	0.083
	test	9	922	1991	2.16	0.104		146	1500	2404	1.60	0.070		14	225	1034	4.60	0.020	
v2	train	10	6954	18968	2.73	0.091	0.088	203	3000	12085	4.03	0.126	0.114	88	2564	10109	3.94	0.165	0.147
	test	10	2923	4863	1.66	0.079		176	2000	5092	2.55	0.095		79	4937	5521	1.12	0.124	
v3	train	11	12078	32150	2.66	0.079	0.069	218	4000	22394	5.60	0.129	0.112	142	4647	20117	4.33	0.133	0.134
	test	11	5084	7470	1.47	0.044		187	3000	9137	3.05	0.087		122	4921	9668	1.96	0.135	
v4	train	9	3861	9842	2.55	0.095	0.086	222	5000	33916	6.78	0.148	0.132	77	2092	9289	4.44	0.157	0.117
	test	9	7208	15157	2.10	0.080		204	3500	14554	4.16	0.107		61	3294	8520	2.59	0.087	

Table 2: Comparison with the baseline methods (AUC-PR).

	WN18RR				FB15k-237				NELL-995			
Methods	**v1**	**v2**	**v3**	**v4**	**v1**	**v2**	**v3**	**v4**	**v1**	**v2**	**v3**	**v4**
Neural-LP	86.02	83.78	62.90	82.06	69.64	76.55	73.95	75.74	64.66	83.61	87.58	85.69
DRUM	86.02	84.05	63.20	82.06	69.71	76.44	74.03	76.20	59.86	83.99	87.71	85.94
RuleN	90.26	89.01	76.46	85.75	75.24	88.70	91.24	91.79	84.99	88.40	87.20	80.52
GraIL	94.34	94.18	85.80	92.72	84.69	90.57	91.68	94.46	**86.05**	92.62	93.34	87.50
LCILP (Ours)	**95.51**	**96.86**	**90.87**	**94.12**	**85.64**	**91.15**	**92.93**	**94.63**	79.23	**94.31**	**94.10**	**89.92**

Table 3: Comparison with the baseline methods (Hits@10).

	WN18RR				FB15k-237				NELL-995			
Methods	**v1**	**v2**	**v3**	**v4**	**v1**	**v2**	**v3**	**v4**	**v1**	**v2**	**v3**	**v4**
Neural-LP	74.37	68.93	46.18	76.13	52.92	58.94	52.90	55.88	40.78	78.73	82.71	80.58
DRUM	74.37	68.93	46.18	76.13	52.92	58.73	52.90	55.88	19.42	78.55	82.71	80.58
RuleN	80.85	78.23	53.39	71.59	49.76	77.82	**87.69**	85.60	53.50	81.75	77.26	61.35
GraIL	82.45	78.68	58.43	73.41	64.15	81.80	82.83	89.28	**59.50**	93.25	91.41	73.19
LCILP (Ours)	**88.30**	**84.12**	**72.68**	**79.46**	**70.97**	**81.89**	84.33	**89.84**	52.40	**93.58**	**92.15**	**82.90**

4.2. Baselines and implementation details

We evaluate our model against the following baselines and state-of-the-art competitors: Neural-LP [31], DRUM [33], RuleN [32], and GraIL [15]. We implement our model on PyTorch [56]. In order to extract the local subgraphs, we set the teleportation probability to a commonly used value of $\alpha = 0.15$ [25, 26] for all the datasets, and the approximation parameter ϵ to 1e-3 for WN18RR, NELL-995 and v1 of FB15k-237, while we set it to 1e-4 for the remaining partitions of FB15k-237 (v2, v3, and v4). The local subgraphs extracted by our model reach 3-hop for FB15k-237 and NELL-995, and 4-hop for WN18RR. However, in order to maintain a fair comparison with GraIL [15], we limit the subgraphs to 3-hop around the target link, and we employ a 3-layer GNN. To train our model, we use Adam optimizer [57] with a learning rate of 0.01, and γ is set to 10. The number of training epochs is set to 50. Finally, we run our experiments on a machine with two Intel Xeon Gold 6230 CPUs running at 2.10 GHz with 128 GB of memory, powered by Nvidia Quadro RTX 5000 GPU with 16 GB of memory. Our code is made available at: `https://github.com/hebatef/LCILP`.

4.3. Evaluation protocol

We evaluate our model following the same protocol used in GraIL [15]. More specifically, we use the area under the precision-recall curve (AUC-PR) and Hits@10, classification and ranking metrics, respectively. To compute AUC-PR, we sample one negative triplet for each test triplet and evaluate which triplet has the larger score. For Hits@10, we rank each test triplet among 50 randomly sampled negative triplets in terms of the scores, to see whether the true triplet can rank in the top 10. The negative triplets are obtained by replacing the head or the tail of the test triplets with other entities.

4.4. Results and discussion

In Table 2 and Table 3, we report the mean AUC-PR and Hits@10 averaged over five runs, respectively. Our model outperforms the strongest baseline (GraIL), on almost all the datasets; 11 out of 12 datasets. In terms of AUC-PR, the performance gain is significant, with an improvement of up to 5.07%. In terms of Hits@10 evaluation metric, again, the improvement is consistent across the datasets with a maximum of 14.25% absolute increase. To answer *RQ*1, in section 4.4.1, we analyze the performance of our model against GraIL, which reason over subgraphs extracted by a fixed number of hops. Moreover, in section 4.4.2, we explore the relationship between the average clustering coefficient of a graph and the performance of our local clustering-based model. In order to answer *RQ*2, we analyze the performance of our model on different *k*-hop settings in section 4.4.3.

Table 4: The maximum number of subgraph nodes under different subgraph extraction strategies (we highlight the percentage decrease).

	WN18RR		FB15k-237		NELL-995	
	GraIL	*Ours*	GraIL	*Ours*	GraIL	*Ours*
v1	172	102 (-40.7%)	1191	116 (-90.3%)	890	218 (-75.5%)
v2	202	109 (-46.0%)	1989	743 (-62.6%)	1813	245 (-86.5%)
v3	514	126 (-75.5%)	3032	622 (-79.5%)	3680	293 (-92.0%)
v4	121	103 (-14.9%)	3951	538 (-86.4%)	2060	135 (-93.4%)

4.4.1. Analysis on different subgraph extraction strategies

On WN18RR datasets, we obtain the highest gain in the performance in comparison to FB15k-237 and NELL-995; the increase in the performance of WN18RR v3 reaches +5.07% and +14.25% in AUC-PR and Hits@10, respectively. WN18RR v3 has the largest reduction in the number of nodes among WN18RR datasets when using our model, as shown in Table 4. This supports the intuition that limiting the subgraphs to a local cluster can bring a high gain in the performance. When taking into consideration FB15k-237, the maximum increases are +1.25% and +6.82% in AUC-PR and Hits@10, respectively. This shows that our proposed model is competitive also when dealing with more challenging datasets that have a high number of unique relation types. Similar to WN18RR v3, FB15k-237 v1 has the largest reduction in the number of nodes and the largest increase in the performance. Finally, in NELL-995, the improvement reaches a maximum of +1.69% and +9.71% in AUC-PR and Hits@10, respectively. Consistent with WN18RR and FB15k-237, NELL-995 v4 has the largest subgraph reduction and the largest increase in performance among NELL-995 datasets.

From these results, we can confirm that sampling fewer but more relevant nodes enhance the performance of link prediction and that relying on local and dense subgraphs brings better performance due to the fact that topology-based heuristics often rely on triangular patterns to extract logical rules [54, 55]. Triangles tend to exist between the target relation and the relations of the surrounding dense subgraph. Moreover, links are more likely to be established between nodes in a tightly connected cluster than randomly. On the other hand, subgraphs induced based on a fixed number of hops often contain noisy relations that degrade the link prediction performance.

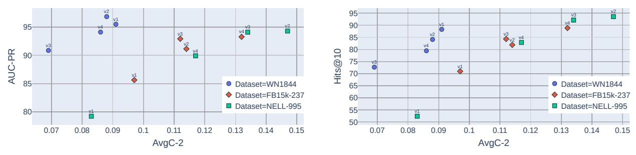


Figure 3: *AvgC-2* versus ICILP performance (AUC-PR and Hits@10). Considering each dataset separately, the higher the *AvgC-2* the better the performance.

4.4.2. Average clustering coefficient and performance

The clustering coefficient is an indicator of the degree of tightness between nodes [58]. Thus, a high average clustering coefficient indicates that there is a high tendency to find local clusters around the target link. To explore the relationship between the average clustering coefficient of a graph and the performance of inductive link prediction, we plot in Figure 3 the clustering coefficient, calculated over the train and test sets (*AvgC-2* in Table 1), and the performance of our model (AUC-PR and Hits@10). In general, as the clustering coefficient increases, there is an improvement in the performance. In graphs with low average clustering coefficient, such as WN18RR v3, FB15k-237 v1, and NELL-995 v1, link prediction performs poorly. This highlights the importance of focusing on local clusters around the target links, and those topology-based heuristics rely on triangles to extract logical rules.

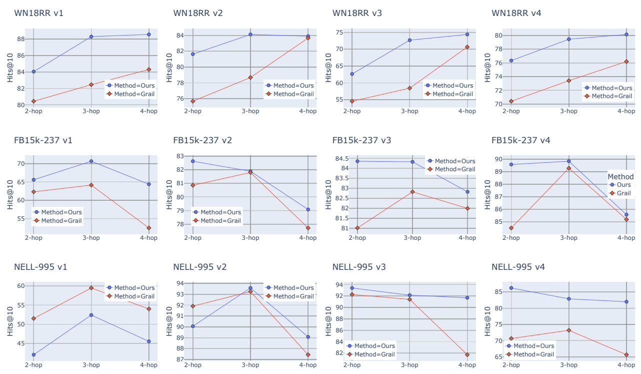


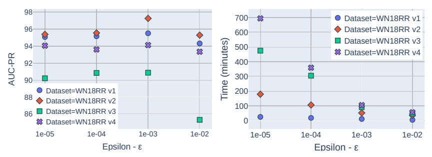
Figure 4: Performance of LCILP (Ours) and GraIL on *k*-hop.

4.4.3. Performance on k-hop

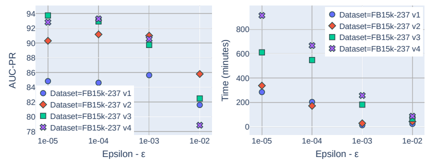
We conduct additional experiments to analyze the performance of link prediction when using 2-hop and 4-hop subgraphs. More precisely, we use PPR for ranking nodes around a target link, but we do not utilize the *conductance* to extract a local cluster and to specify the maximum number of hops. We instead fix k, and extract the subgraph by inducing nodes with the highest PPR scores up to k-hop. We employ 2-layer and 4-layer R-GCNs, respectively, for the different k-hop settings. As shown in Figure 4, we observe that 4-hop frequently degrade the performance of link prediction for the graphs with high average node degree, such as FB15k-237 and NELL-995. These graphs tend to have large subgraphs around the target relation. This confirms that larger subgraphs can be noisy to predict a link and that focusing on the local cluster enhances the link prediction performance. Finally, from the reported results, we conclude that our proposed model still outperforms GraIL on most of the datasets over the different k-hop settings.

4.5. Parameter sensitivity

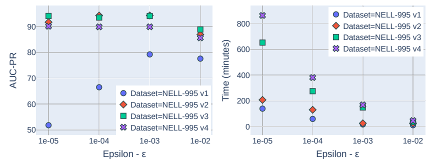
To analyze the effect of ϵ hyperparameter on the performance of our model, we experiment with different values of $\epsilon = [1e-2, 1e-3, 1e-4, 1e-5]$. The results show a trade-off between performance and computational time; AUC-PR increases for a more accurate approximation of the PPR vectors (smaller ϵ), which in turn corresponds to higher computational time. For example, in Figure 5a, WN18RR v3 model with $\epsilon = 1e-2$ has an overall (training + testing) runtime of 46 minutes and AUC-PR = 85.32, while a model with $\epsilon = 1e-3$ has an overall runtime of 91 minutes and AUC-PR = 90.87. In Figure 5b, FB15k-237 v3 model with ϵ = 1e-2, has an overall



(a) WN18RR.



(b) FB15k-237.



(c) NELL-995.

Figure 5: ϵ versus AUC-PR and computational time of LCILP on different datasets.

runtime of 66 minutes and AUC-PR = 82.48, while a model with $\epsilon = 1e-4$ has an overall runtime of 182 minutes and AUC-PR = 92.93. We observe the same behavior in NELL-995 v3; for $\epsilon = 1e-2$, the computational time is 35 minutes with a performance of AUC-PR = 88.87, while for $\epsilon = 1e-3$, the runtime reaches to 148 minutes for AUC-PR = 94.10, as shown in Figure 5c. However, for NELL-995 v1 dataset, we observe that the algorithm converges at $1e-3$, and that lower values of ϵ, such as $1e-5$, do not result in an increase in the performance. We consider this trade-off to select different values for ϵ for the different datasets as specified in the implementation details. Although our method consumes additional time for extracting PPR-based local subgraphs, the time complexity of GNN decreases, since the extracted local subgraphs are smaller than the ones from GraIL.

4.6. Case study on hub nodes

We perform additional analysis to understand if the local subgraphs extracted by our model contain contextual and relevant information. In particular, we validate if our proposed subgraph extraction method can exclude hub nodes (and irrelevant nodes connected to them). Hub nodes are nodes with a very high node degree. They are usually connected to most of the nodes in the graph. Since hub nodes connect different clusters of the graph, they usually have low clustering coefficient [18]. For instance, in the train set of FB15k-237 v1, there is a hub node reflecting Freebase entity *'/m/02h40lc'*, which represents *'English'* (the language used by almost all the movies in the dataset). This entity has in-degree = 265 and clustering coefficient = $9.96e-4$. To ensure that such a node does not exist in the subgraphs extracted by our model, we calculate the maximum node degree in the subgraphs extracted using our proposed method versus the ones extracted with GraIL. We find that the maximum node degree with GraIL is = 265, while it is = 15 using our approach. This shows that our subgraph extraction method is able to filter out hub nodes and the other irrelevant nodes connected to them.

5. Conclusions and future work

We have proposed a novel strategy for inductive link prediction in Knowledge Graphs. Unlike reasoning over subgraphs extracted by a fixed number of hops, we sampled tightly-related subgraphs around the target links using a PPR-based local clustering method, and then applied a GNN for reasoning over the extracted subgraphs . Experimental results demonstrated the effectiveness of our proposed approach. Moreover, we showed that the performance of link prediction tends to increase with the increase of the average clustering coefficient of the graph and that graphs with high average node-degree require a small number of hops for effective performance on the link prediction task. In the future, we can combine any state-of-the-art prediction method with our scheme. Additionally, our model can be extended with learnable weights that control the amount of information from neighbors based on the topological properties of the graph.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 870743

References

[1] J. Berant, A. Chou, R. Frostig, P. Liang, Semantic parsing on freebase from question-answer pairs, in: Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1533–1544.

[2] L. Heck, D. Z. Hakkani-Tür, G. Tür, Leveraging knowledge graphs for web-scale unsupervised semantic parsing, in: Proc. Interspeech 2013, 2013, pp. 1594–1598. doi:10.21437/Interspeech.2013-401.

[3] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, D. S. Weld, Knowledge-based weak supervision for information extraction of overlapping relations, in: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, 2011, pp. 541–550.

[4] J. Daiber, M. Jakob, C. Hokamp, P. N. Mendes, Improving efficiency and accuracy in multilingual entity extraction, in: Proceedings of the 9th International Conference on Semantic Systems, 2013, pp. 121–124.

[5] Y. Fang, K. Kuan, J. Lin, C. Tan, V. R. Chandrasekhar, Object detection meets knowledge graphs, in: IJCAI, 2017, pp. 1661–1667.

[6] A. Zareian, S. Karaman, S.-F. Chang, Bridging knowledge graphs to generate scene graphs, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), Computer Vision – ECCV 2020, Springer International Publishing, Cham, 2020, pp. 606–623.

[7] A. Bordes, J. Weston, N. Usunier, Open question answering with weakly supervised embedding models, in: Joint European conference on machine learning and knowledge discovery in databases, Springer, 2014, pp. 165–180.

[8] X. Huang, J. Zhang, D. Li, P. Li, Knowledge graph embedding based question answering, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 105–113.

[9] M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs, Proceedings of the IEEE 104 (1) (2015) 11–33.

[10] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Advances in neural information processing systems 26 (2013).

[11] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, Proceedings of the AAAI Conference on Artificial Intelligence 28 (1) (Jun. 2014). doi:10.1609/aaai.v28i1.8870.

[12] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: International conference on machine learning, PMLR, 2016, pp. 2071–2080.

[13] Z. Sun, Z. Deng, J.-Y. Nie, J. Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, ArXiv abs/1902.10197 (2019).

[14] M. Zhang, Y. Chen, Link prediction based on graph neural networks, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 31 of NIPS'18, Curran Associates, Inc., 2018, p. 5171–5181.

[15] K. Teru, E. Denis, W. Hamilton, Inductive relation prediction by subgraph reasoning, in: International Conference on Machine Learning, PMLR, 2020, pp. 9448–9457.

[16] J. Chen, H. He, F. Wu, J. Wang, Topology-aware correlations between relations for inductive link prediction in knowledge graphs, Proceedings of the AAAI Conference on Artificial Intelligence 35 (7) (2021) 6271–6278.

[17] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE transactions on neural networks 20 (1) (2008) 61–80.

[18] T. G. Lewis, Network science: Theory and applications, John Wiley & Sons, 2011.

[19] K. Oono, T. Suzuki, Graph neural networks exponentially lose expressive power for node classification, arXiv: Learning (2020).

[20] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, X. Sun, Measuring and relieving the over-smoothing problem for graph neural networks from the topological view, Proceedings of the AAAI Conference on Artificial Intelligence 34 (04) (2020) 3438–3445. doi:10.1609/aaai.v34i04.5747.

[21] Z. Chen, L. Chen, S. Villar, J. Bruna, Can graph neural networks count substructures?, ArXiv abs/2002.04025 (2020).

[22] G. Bouritsas, F. Frasca, S. Zafeiriou, M. M. Bronstein, Improving graph neural network expressivity via subgraph isomorphism counting, IEEE transactions on pattern analysis and machine intelligence PP (2022).

[23] M. Curado, Return random walks for link prediction, Information Sciences 510 (2020) 99–107.

[24] A. Bojchevski, J. Klicpera, B. Perozzi, A. Kapoor, M. Blais, B. Rózemberczki, M. Lukasik, S. Günnemann, Scaling graph neural networks with approximate pagerank, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 2464–2473.

[25] A. Vattani, D. Chakrabarti, M. Gurevich, Preserving personalized pagerank in subgraphs, in: Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11, Omnipress, Madison, WI, USA, 2011, p. 793–800.

[26] J. Leskovec, C. Faloutsos, Sampling from large graphs, in: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 2006, pp. 631–636.

[27] H. Nassar, K. Kloster, D. F. Gleich, Strong localization in personalized pagerank vectors, in: International Workshop on Algorithms and Models for the Web-Graph, Springer, 2015, pp. 190–202.

[28] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 1025–1035.

[29] S. Yang, B. Hu, Z. Zhang, W. Sun, Y. Wang, J. Zhou, H. Shan, Y. Cao, B. Ye, Y. Fang, et al., Inductive link prediction with interactive structure learning on attributed graph, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2021, pp. 383–398.

[30] P. Wang, J. Han, C. Li, R. Pan, Logic attention based neighborhood aggregation for inductive knowledge graph embedding, 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, AAAI Press, 2019, pp. 7152–7159. doi:10.1609/aaai.v33i01.33017152.

[31] F. Yang, Z. Yang, W. W. Cohen, Differentiable learning of logical rules for knowledge base reasoning, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 2316–2325.

[32] C. Meilicke, M. Fink, Y. Wang, D. Ruffinelli, R. Gemulla, H. Stuckenschmidt, Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion, in: International semantic web conference, Springer, 2018, pp. 3–20.

[33] A. Sadeghian, M. Armandpour, P. Ding, D. Wang, Drum: End-to-end differentiable rule mining on knowledge graphs, in: NeurIPS, 2019, pp. 15347–15357.

[34] M. Zhang, P. Li, Y. Xia, K. Wang, L. Jin, Labeling trick: A theory of using graph neural networks for multi-node representation learning, Advances in Neural Information Processing Systems 34 (2021).

[35] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: European semantic web conference, Springer, 2018, pp. 593–607.

[36] Q. Lin, J. Liu, Y. Pan, L. Zhang, X. Hu, J. Ma, Rule-enhanced iterative complementation for knowledge graph reasoning, Information Sciences 575 (2021) 66–79.

[37] Y. Zhang, X. Chen, Y. Yang, A. Ramamurthy, B. Li, Y. Qi, L. Song, Efficient probabilistic logic reasoning with graph neural networks, arXiv preprint arXiv:2001.11850 (2020).

[38] L. V. Harsha Vardhan, G. Jia, S. Kok, Probabilistic logic graph attention networks for reasoning, in: Companion Proceedings of the Web Conference 2020, 2020, pp. 669–673.

[39] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, V. Prasanna, Graphsaint: Graph sampling based inductive learning method, ArXiv abs/1907.04931 (2020).

[40] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.-J. Hsieh, Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 257–266.

[41] Y. Rong, W. Huang, T. Xu, J. Huang, Dropedge: Towards deep graph convolutional networks on node classification, in: ICLR, 2020.

[42] J. Klicpera, A. Bojchevski, S. Günnemann, Predict then propagate: Graph neural networks meet personalized pagerank, in: ICLR, 2019.

[43] R. Andersen, F. Chung, K. Lang, Local graph partitioning using pagerank vectors, in: 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), IEEE, 2006, pp. 475–486.

[44] R. Andersen, K. J. Lang, Communities from seed sets, in: Proceedings of the 15th international conference on World Wide Web, 2006, pp. 223–232.

[45] Y. Yan, Y. Bian, D. Luo, D. Lee, X. Zhang, Constrained local graph clustering by colored random walk, in: The world wide web conference, 2019, pp. 2137–2146.

[46] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, ArXiv abs/1810.00826 (2019).

[47] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, M. Gamon, Representing text for joint embedding of text and knowledge bases, in: Proceedings of the 2015 conference on empirical methods in natural language processing, 2015, pp. 1499–1509.

[48] G. A. Miller, Wordnet: a lexical database for english, Communications of the ACM 38 (11) (1995) 39–41.

[49] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings, in: AAAI, 2018, pp. 1811–1818.

[50] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008, pp. 1247–1250.

[51] W. Xiong, T. Hoang, W. Y. Wang, DeepPath: A reinforcement learning method for knowledge graph reasoning, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 564–573. doi:10.18653/v1/D17-1060.

[52] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, T. M. Mitchell, Toward an architecture for never-ending language learning, in: AAAI, 2010, pp. 1306–1313.

[53] D. J. Watts, S. H. Strogatz, Collective dynamics of 'small-world'networks, nature 393 (6684) (1998) 440–442.

[54] M. R. Douglas, M. Simkin, O. Ben-Eliezer, T. Wu, P. Chin, T. V. Dang, A. Wood, What is learned in knowledge graph embeddings?, in: Complex Networks & Their Applications X, Springer International Publishing, Cham, 2022, pp. 587–602.

[55] H. Naacke, O. Curé, Triag, a framework based on triangles of rdf triples, in: Proceedings of The International Workshop on Semantic Big Data, 2020, pp. 1–6.

[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Advances in neural information processing systems 32 (2019) 8026–8037.

[57] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR abs/1412.6980 (2015).

[58] M. Nie, D. Chen, D. Wang, Graph embedding method based on biased walking for link prediction, Mathematics 10 (20) (2022) 3778.