Using Graph Algorithms to Pretrain Graph Completion Transformers

Jonathan Pilault1*, Michael Galkin1, Bahare Fatemi2*, Perouz Taslakian3*, David Vasquez4, Christopher Pal1,4,5

1Polytechnique Montreal & Mila, 2 Google Research, 3Samsung AI, 4ServiceNow Research, 5Canada CIFAR AI Chair

{jonathan.pilault,michael.galkin,christopher.pal}@mila.quebec

Abstract

Recent work on Graph Neural Networks has demonstrated that self-supervised pretraining can further enhance performance on downstream graph, link, and node classification tasks. However, the efficacy of pretraining tasks has not been fully investigated for downstream large knowledge graph completion tasks. Using a contextualized knowledge graph embedding approach, we investigate five different pretraining signals, constructed using several graph algorithms and no external data, as well as their combination. We leverage the versatility of our Transformer-based model to explore graph structure generation pretraining tasks (i.e. path and k-hop neighborhood generation), typically inapplicable to most graph embedding methods. We further propose a new path-finding algorithm guided by information gain and find that it is the best-performing pretraining task across three downstream knowledge graph completion datasets. While using our new path-finding algorithm as a pretraining signal provides 2-3% MRR improvements, we show that pretraining on all signals together gives the best knowledge graph completion results. In a multitask setting that combines all pretraining tasks, our method surpasses the latest and strong performance on downstream graph, link, and node classification tasks. In an arity-2 graph, a KG can be viewed as a set of triplets \( G = \{ (h_i, r_i, t_i) \}, i \in \{1, ... , T\} \), where \( h_i \) is the head entity, \( t_i \) is the tail entity, and \( r \) is the relation between \( h_i \) and \( t_i \). We can feed the sequence \( h_i, r_i, [\text{mask}] \) or \( [\text{mask}], r_i, t_i \), where \([\text{mask}]\) represents the entity to predict. Since Transformers can process sequences of arbitrary length, we can go beyond triple-based KGs.

1 Introduction

Transfer learning has emerged as a powerful technique in several domains [Ramachandran, Liu, and Le 2017; Pennington, Socher, and Manning 2014; Tomar et al. 2017; Subramanian et al. 2018; Oquab et al. 2014; Yosinski et al. 2014; Trinh, Luong, and Le 2019], in which a model is first pretrained on relevant tasks before being fine-tuned on a downstream task. In most modern vision and NLP applications, such pretraining is often based on the versatile Transformer model [Vaswani et al. 2017] using self-supervised learning on unlabeled data [Devlin et al. 2019; Raffel et al. 2020; Bao et al. 2022]. Transformer-based and self-supervised pretraining have also been applied in the graph representation learning scenarios. These studies, however, focus mostly on non-relational graphs [Thakoor et al. 2022; Fatemi, El Asri, and Kazemi 2021] or small molecular graphs [Ying et al. 2021], leaving pretraining approaches for Knowledge Graphs (KG) link prediction relatively unexplored.

Many Graph Neural Network (GNN) techniques use positional embeddings of the entities in the relation along with entity/relation embeddings to represent tuples [You, Ying, and Leskovec 2019]. Transformer Language Models [Vaswani et al. 2017] use positional embeddings in a similar way. Transformers can perform contextualized link prediction by masking out one of the entities in an input triple. In an arity-2 graph, a KG can be viewed as a set of triplets \( G = \{ (h_i, r_i, t_i) \}, i \in \{1, ... , T\} \), where \( h_i \) is the head entity, \( t_i \) is the tail entity, and \( r \) is the relation between \( h_i \) and \( t_i \). We can feed the sequence \( h_i, r_i, [\text{mask}] \) or \( [\text{mask}], r_i, t_i \), where \([\text{mask}]\) represents the entity to predict. Since Transformers can process sequences of arbitrary length, we can go beyond triple-based KGs. We show that our approach can also be applied to the more general case of Knowledge Hypergraphs (KHGs, Fatemi et al.), where edges are composed of tuples with an arbitrary number of entities. It would require many modifications to apply KHGs to other KG Embedding methods [Fatemi et al. 2020]. With the versatility of our base model, we can also perform other tasks, such as autoregressive path generation or query conditioned neighborhood prediction, that GNNs or KG embeddings cannot.

So far, the most common pretraining approach to KG link prediction is to use a Transformer pretrained on a large corpora of text, such as Wikipedia [Yao, Mao, and Luo 2019; Wang et al. 2021; Li, Yi, and He 2022]. There are two issues with using pretrained language models: (1) Knowledge or information leakage may occur from the pretraining text data to common KG completion benchmarks such as FB15k-237 (Bollacker et al. 2008), extracted from Wikidata (formerly Freebase) and WN18RR (Dettmers et al. 2018), extracted from Wordnet (Miller 1993); (2) If the pretraining text corpora is out of domain with respect to the test KG, it is unclear if external data unrelated to the targeted knowledge base will be helpful for knowledge completion. Further, not using external data is instrumental in situations where data is scarce, confidential, or classified such as medical records, crime networks, or robotics [Wang, Zhang, and Liao 2019]. It may be infeasible to pretrain a Transformer on a more closely re-
We evaluate the effect of pretraining strategies on triple-IP.

We introduce a new path-finding algorithm (IP) that is guided by the Kullback-Leibler divergence (i.e. information gain) in Section 4.2. Our techniques can also be applied to a generalization of KGs, i.e. KHGs.

Our contributions are as follows:

1. We report results of five pretraining objectives suitable for our Transformer-based contextualized KG embedding method in Section 4.

2. We introduce a new path-finding algorithm (IP) that is guided by the Kullback-Leibler divergence (i.e. information gain) in Section 4.2. Our techniques can also be applied to a generalization of KGs, i.e. KHGs.

3. We evaluate the effect of pretraining strategies on triple-based KGs as well as on KHGs and show that our new path-based (IP) and our multitask (ALL) pretraining schemes perform best for both low and high density knowledge graphs in Section 4.

2 Related Works

Our method is most related to Contextualized KG Embeddings [Wang et al. 2019] and Self-Supervised Graph Representation pretraining. We will discuss the two lines of work below.

Knowledge Graph Embedding methods encode distributed representations of entities and relations using a continuous vector. The representation is learned via a scoring function that measures the plausibility that a given triple \( \{ (h_i, r_i, t_i) \} \) exists. Such embeddings maximize the plausibility of observed triples, which may not be predictive enough for downstream tasks [Wang, Wang, and Guo 2015]. To make the embeddings more transferable, researchers have incorporated other types of information. For example, external data sources can complement a KG embedding by also encoding entity types [Guo et al. 2015] [Xie, Liu, and Sun 2016] or textual descriptions [Xie et al. 2016] [Wang and Li 2016]. As explored here, we can also enhance KG embedding by only using structural features from the graph. For example, relational paths and multi-hop relationships between entities (see section 4 for more details), are a type of structural information that has proven useful [Lin et al. 2015] [Toutanova et al. 2016] [Das et al. 2017] for KG completion.

Self-Supervised Graph Representation learning techniques broadly fall into three categories and include methods that: (1) use random walk procedures to encode diverse neighborhoods [Perozzi, Al-Rfou, and Skiena 2014] [Grover and Leskovec 2016]; (2) reconstruct a graph’s adjacency matrix [Kipf and Welling 2016] [Hasanzadeh et al. 2019]; (3) maximize mutual information between local node representations and global graph representations [Veličković et al. 2019] [You et al. 2020] [Liang et al. 2021]. For applications in chemistry and biology, most pretraining techniques also use additional datasets and signals like MAE loss [Hu et al. 2020b] [Ying et al. 2021], however, our technique does not use external data. With the exception of Deep Relational Graph Infomax (DRGI) [Liang et al. 2021], most self-supervised pretraining schemes based on graph structure have never been applied to solve large KG completion tasks. Other techniques such as KG-BERT [Yao, Mao, and Luo 2019] use a BERT-based [Devlin et al. 2019] pretrained language models with entity descriptions for KG link prediction. Again, such work relies on external data and additional information (descriptions) while our pretraining signals are self-contained.

3 Definition and Notation

In this section, we provide definitions and background on link prediction, paths, multi-hop neighborhoods and graph adjacency matrices. We also introduce important notations used in equations and algorithms.

Since we are also evaluating our technique on the KHG JF17K, a generalization of KG with arity-n, we provide in this section definitions based on hypergraphs. Without loss of generality, in this section, we formulate our definitions and notation based on hypergraphs. Given a finite set of entities \( E \) and a finite set of relations \( R \), a tuple is an ordered set of the form \( r(e_1, \ldots, e_n) \), where \( r \in R, e_i \in E \) for all \( i = 1, \ldots, n \) and \( |r| = n \) is its arity. Let \( \mathcal{G} \) be the set of ground truth tuples; that is, it specifies all of the tuples that are true so that if a tuple is not in \( \mathcal{G} \), it is false. A knowledge hypergraph (KHG) consists of a subset of the tuples \( \mathcal{G}' \subseteq \mathcal{G} \). A knowledge graph is a special case of a knowledge hypergraph where all relations have arity 2. We let \( E(r) \subseteq E \) denote the set of all entities that appear in a tuple in \( \mathcal{G}' \) having relation \( r \). That is, \( E(r) = \{ e_i \mid r(e_1, \ldots, e_i, \ldots) \in \mathcal{G}' \} \). Similarly, we let \( R(e_i) \subseteq R \) denote the set of all relations that appear in a tuple in \( \mathcal{G}' \) linked to entity \( e_i \). That is,
\( R(e_i) = \{ r| r(e_1, \ldots, e_n) \in G' \} \). We say that two tuples are incident in \( G' \) if they share at least one entity. Two entities are connected in \( G' \) if they appear together in a tuple. For example, \( r_1(e_1, e_2, e_3) \) is incident to \( r_2(e_4, e_3, e_5) \) because they share \( e_3 \). Entities \( e_2 \) and \( e_3 \) are connected because they both appear in the first tuple. A query for the knowledge completion task is a tuple with one missing entity that needs to be predicted. We let \( Q_r = [r_1, e_1, \ldots, [\text{mask}], \ldots, e_n] \) denote a query tuple with relation \( r \), where [mask] is the placeholder token (the masked-out entity) for the entity we want to predict.

A path \( P \) in a KHG is a sequence of tuples where two consecutive tuples share at least one entity. We say that \( P \) connects entities \( e_i \) and \( e_j \) if the first tuple in \( P \) contains \( e_i \) and the last tuple contains \( e_j \). A relational path \( P_{rel} \) between \( e_i \) and \( e_j \) is the sequence of relations along the edges of a path connecting \( e_i \) and \( e_j \) such that no relation is repeated along the path (without cycles). For example, \( e_1 \) and \( e_6 \) are connected through path \( P : r_1(e_1, e_2, e_3), r_2(e_4, e_5, e_3), r_3(e_4, e_5, e_6), r_2(e_4, e_5, e_3), r_3(e_4, e_5, e_6) \), and have a relational path \( P_{rel} : r_1, r_2, r_3 \). Given the set of entities in queries \( \{Q_r\}_r \), we define the set of all possible paths between pairwise entities as \( \{P_{rel}\}_r \).

The \( k \)-hop neighborhood of entity \( e_i \), denoted \( N_k(e_i) \), is the unordered set of entities \( E_{N_k} = \{e_k, \ldots, e_j\} \subseteq E \) enclosed in a \( k \)-hop radius around \( e_i \). We denote the relationless adjacency matrix of entities in \( N_k(e_i) \) as \( A_{E_{N_k}} \in \mathbb{R}^{|E_{N_k}| \times |E_{N_k}|} \). For a relational graph \( G' \), we define a relationless adjacency matrix \( A_E \) as:

\[
A_E = \sum_{r \in R} A_{E_{(r)}}, A_E \in \mathbb{R}^{|E| \times |E|}.
\]

Adjacency \( A \) has a certain order of appearance of entities in columns and rows which is arbitrarily set by the order in which dataset tuples are sampled. We let \( \tilde{A} \) be the equivalent adjacency matrix with a permuted order of entities in columns and rows (e.g.: \( r_1, r_2, r_3, r_4 \) can become \( r_2, r_3, r_1, r_4 \)). The local clustering coefficient \( c_{e_i} \) (Watts and Strogatz 1998) of \( N_k(e_i) \) measures the proportion of closed triangles in the local \( k \)-hop neighborhood of an entity such that:

\[
c_{e_i} = \frac{\binom{|e_1, e_2 \in E: e_1, e_2 \in N_k(e_i)|}{2}}{d_{e_i}}.
\]

where \( d_{e_i} = \sum_{e_j \in N_k(e_i)} A[e_i, e_j] \) is the node degree of \( e_i \). For the pretraining task in section 4.3, we define the probability that an entity \( e_i \) is found in a \( k \)-hop neighborhood (probability of occurrence) \( O(e_i) \) as:

\[
O(e_i) = \frac{d_{e_i}}{\sum_{e_j \in E_{N_k}} d_{e_j}}, \quad O_{N_k} = \begin{bmatrix} O(e_1) \\ \vdots \\ O(e_{e_1}) \end{bmatrix}
\]

4 Methodology

In this section, we provide the motivation and details of the five different graph algorithms that we employed to create pretraining tasks for our experiments: (1) Relational Shortest Path sequence generation (SP); (2) Information gain Path sequence generation (IP); (3) K-Hop Neighbor prediction (KHN); (4) Invariant Adjacency matrix classification (IVA); (5) Local Clustering Coefficient estimation (LCC). Please see Section 3 for more details on definitions and notation used in this section. When jointly trained (ALL), we use all pretraining tasks and prepend a task token to delineate each task. Note that for ALL, the total is the sum of each pretraining task’s loss. An overview of the task objectives is outlined in Table 1.

| Task Input | Target | Objective | Type |
|------------|--------|-----------|------|
| SP \( \{e_i, r, e_j\} \) | \( P_{rel} \) \( \sum_{k} f(p_k | p_{<k}, e_i, r, e_j) \) | SG |
| Neighborhood Based | | | |
| KHN \( \{e_i, E_{N_k-1}\} \) \( O_{N_k} \) \( KL(O_{N_k}||f(e_i, E_{N_k-1})) \) | MP |
| IVA \( \{A, \tilde{A}|\tilde{A}'\} \) \( \{|0|\} \) | \( yklag(f(A, \tilde{A}|\tilde{A}')\) | BC |
| LCC \( \{e_i, E_{N_k-1}\} \) \( c_{e_i} \) \( (f(e_i) - c_{e_i})^2_k \) | R |

Table 1: Overview of Graph Algorithm Pretraining Tasks. Rel=Uses Relations; SG=Sequence Generation; BC=Binary Classification; MP=Multi-label Prediction; R=Regression.

4.1 Relational Shortest Paths (SP)

Using the definition of global and quasi-local graph structural information from (Benhidour, Almeshkhas, and Kerrache 2022), path signals capture global graph structure since all paths cross each node in the graph at least once. The idea to use paths is loosely inspired by global methods such as the Katz index that considers sets of paths between two query nodes to measure the likelihood of a link between them (Katz 1953). The set of relational paths \( \{P_{rel}\}_r \), as defined in Section 3, may yield an exponential number of choices. In practice, we would like to limit the number of such paths and hence, we need a way to select a subset of \( \{P_{rel}\}_r \). There are a few possible heuristics for selecting a subset of such paths. For example, we can find shortest paths based on Dijkstra’s algorithm and then just keep the sequences of relations joining two entities \( \{e_i, r, e_j\} \). As specified in Table 1, we condition our path sequence generation with \( \{e_i, r, e_j\} \). The next token generated is the first relation on a sampled path \( \{P_{rel}\}_r \). Our path-based pretraining algorithms allow the model to explore \( G \) beyond the tuples included in \( G' \) since \( G \) includes relational paths between entities that may not appear in any of the train, validation, or test sets. When no path exist between \( \{e_i, e_j\} \), the model target is the [no path] token. We posit that the quality of the paths is also important and created a new path finding algorithm called IP to test this hypothesis.

4.2 Information gain Paths (IP)

While SP provides a measure of subgraph distance and connectivity, i.e. the number of elements between entities \( \{e_i, e_j\} \) in the graph, it may often yield reasoning chains having little semantic overlap with the original query. For example, a masked query “[Spain] [form of government]"
Algorithm 1: Top-k Path Information Gain

**Input:** Training set $G$; query relation $r$; $k$ number of top paths; max relational paths length (max hops) $l$

**Output:** at most $n = k^l$ relational paths $\{P_{rel}\}$.

Given a query $r$, we first collect all other relations that have the highest information gain for the paths constructed so far.

Algorithm 2: Finding all relations incident to $r$ in $G$.

**Function** FindIncidentRelations($r$, $P_{rel}$):

1. $R' \leftarrow \emptyset$.
2. for $r' \in R \setminus \{r\}$ do
3. if $E(r') \cap E(r) \neq \emptyset$ and $r' \notin P_{rel}$ then
4. $R' \leftarrow r'$;
5. return $R'$.

Algorithm 3: Finding $r \in R$ with top $k$ entropy $H(r)$ or with $k$ conditional entropy $H(r|r').$

**Function** TopEntropy($k$, $R'$):

1. Compute $H(r')$ for all the relations $r' \in R'$ (Eq. 3).
2. return $r'$ with top-$k$ highest $H(r')$.

**Function** BottomCondEntropy($k$, $R''$, $r'$):

1. Compute $H(r''|r')$ for all $r'' \in R''$ (Eq. 5).
2. return $r''$ with lowest $H(r''|r')$.

[mask]” produces many SP such as “[nominated for], [film/country]”, which is a sequence of relations unrelated to “[form of government]”. Further, SP algorithms select paths that minimize the distance (number of hops) between two entities. It is therefore formed from shortest paths of the same length, which may limit subgraph exploration. We propose a new path finding algorithm, Information Gain Paths in Algorithm 1. IP has the same initial condition $\{e_i, r, e_j\}$ to start generating the paths and has the same learning objective as SP. IP also helps us reduce the number of paths by selecting the ones that constitute a beneficial context for answering the query based on a measure of information gain. Given a query tuple $Q_r$ (having relation $r$), the algorithm progressively builds relational paths $\{P_{rel}\}$, starting from a relation $r$ such that, at each step, it selects the (at most) $k$ incident relations that would yield the $k$ highest information gain for the paths constructed so far.

With infinite $k$, the algorithm is identical to the breadth-first search as no relations are ignored. Algorithm 1 bears some similarities with beam search (Goldberg and Reddy 1977). There are however a few differences: (1) for $l$ hops and $k$ beam size, we obtain a maximum of $k^l$ paths; (2) a relation already in $P_{rel}$ cannot be used to form the next hop; (3) the paths are formed via a back-chaining process starting from the last relation on path $P_{rel}$ that connects $e_i$ and $e_j$; (4) an extra step is performed to select paths in $\{P_{rel}\}$, linking entities in $Q_r$.

We let $IG(P_{rel})$ denote the Information Gain of relational path $P_{rel}$: $r_1, \ldots, r_l \in G'$ having length $l$ as $IG(P_{rel}) = H(r_1) - \sum_{i=1}^{l-1} H(r_{i-1}|r_{i-1+1})$. We define the Information Entropy $H(r_1)$ for a relation $r_1$, and entities $E(r_1)$ that appear in a tuple in $G'$ having relation $r_1$ as:

$$H(r_1) = -\frac{|E(r_1)|}{|E|} \log \left( \frac{|E(r_1)|}{|E|} \right) + \frac{|E| - |E(r_1)|}{|E|} \log \left( \frac{|E| - |E(r_1)|}{|E|} \right).$$

where $E$ is all entities in $G'$.

The Conditional Information Entropy $H(r_{i-1}|r_i)$ of two consecutive relations is defined as:

$$H(r_{i-1}|r_i) = -\frac{|E(r_{i-1})|}{|E(r_i)|} \log \left( \frac{|E(r_{i-1})|}{|E(r_i)|} \right) \left( U \log(U) + V \log(V) \right)$$

where $U = \frac{|E(r_{i-1})| \cap E(r_i)}{|E(r_{i-1})|}$ and $V = \frac{|E(r_{i-1})| \cap E(r_i)}{|E(r_i)|}$.

Algorithm 1 can be summarized in a four step process that bear resemblance with information gain criterion in decision trees:

1. Starting a query relation $r$, we first collect all other relations that are incident in the graph with $FindIncidentRelations$.
2. We then choose the next $k$ most probable and informational relations (excluding $r$) given information entropy in Equation 4 and using the $TopEntropy$ function.
3. We then choose the next $k$ relations that have the highest information value with respect to relations in the previous step, i.e. that have the lowest conditional information entropy from Equation 5 using the $BottomCondEntropy$ function.
4. Repeat step 3 until we reach $l$ hops.

### 4.3 k-Hop Neighborhood prediction (KHN)

Our next graph algorithm to create our pretraining data is based on the observation that the representation of nodes in a neighborhood $N_k(e_i)$ encodes subgraph structure and provides a more powerful representation of $e_i$ (Hamilton, Ying, and Leskovec 2017b). KHN is a quasi-local graph structural information since we are trying
Table 2: Link prediction test results. H@=Hit@. Results from: [Wang et al. In the table, bold is best and underline is second best. *Uses external data and is a contextualized KG embedding method.**] Methods that require too many modifications to work with hypergraphs (JF17K) were not evaluated.

|                | Pre-Trained | FB15K-237            | WN18RR | JF17K             |
|----------------|-------------|----------------------|--------|-------------------|
|                | MRR | H@1 | H@3 | H@10 | MRR | H@1 | H@3 | H@10 | MRR | H@1 | H@3 | H@10 |
| KG-TRSF 1      | ✓ 0.364 | 0.272 | 0.400 | 0.549 | 0.484 | 0.450 | 0.496 | 0.553 | 0.532 | 0.445 | 0.561 | 0.687 |
| KG-TRSF SP     | ✓ 0.368 | 0.275 | 0.405 | 0.561 | 0.489 | 0.452 | 0.499 | 0.566 | 0.537 | 0.450 | 0.569 | 0.703 |
| KG-TRSF IP (ours) | ✓ 0.372 | 0.277 | 0.412 | 0.579 | 0.491 | 0.454 | 0.507 | 0.575 | 0.549 | 0.457 | 0.582 | 0.723 |
| KG-TRSF KHN    | ✓ 0.365 | 0.273 | 0.401 | 0.550 | 0.484 | 0.449 | 0.497 | 0.556 | 0.536 | 0.450 | 0.565 | 0.701 |
| KG-TRSF LCC    | ✓ 0.364 | 0.272 | 0.400 | 0.549 | 0.486 | 0.451 | 0.498 | 0.563 | 0.532 | 0.445 | 0.561 | 0.679 |
| KG-TRSF IVA    | ✓ 0.365 | 0.272 | 0.401 | 0.555 | 0.483 | 0.448 | 0.497 | 0.560 | 0.538 | 0.453 | 0.568 | 0.705 |
| KG-TRSF ALL    | ✓ 0.380 | 0.279 | 0.422 | 0.591 | 0.499 | 0.456 | 0.511 | 0.588 | 0.554 | 0.465 | 0.594 | 0.715 |

**Contextualized KG Embedding Results**

**SOTA KG Embedding Results**

- BOXE ✓ 0.337 0.238 0.374 0.538 0.451 0.400 0.472 0.541 0.553 0.467 0.596 0.711
- DRGI ✓ 0.362 0.270 0.399 0.549 0.479 0.445 0.496 0.543 — — —
- STAR* ✓ 0.358 0.205 0.322 0.482 0.401 0.243 0.491 0.675 — — —
- LP-BERT* ✓ 0.310 0.223 0.336 0.490 0.482 0.343 0.563 0.752 — — —

4.4 Invariant Adjacency matrix (IVA)

One of the key issues with previous heuristics in Section 4.3 is that a certain order is assumed. However, graph representations are more powerful when they are order invariant (Khasanova and Frossard 2017). To allow order invariance, we propose a binary classification task based on adjacency matrices $E_{N_k}$ of the local neighborhood subgraph. The label is 1 if the Transformer is given inputs \{A, \hat{A}\}, where \hat{A} is the equivalent adjacency matrix to A where the entity order is randomly permuted. The label is 0 if the Transformer is given inputs \{A, A'\}, where A' is a corrupted adjacency matrix. We corrupt the adjacency matrix by randomly swapping the columns or by randomly assigning different adjacency matrix values. Since A is a symmetric matrix, the input to the Transformer is a flattened sequence of the upper triangular matrix that contains (1) column entities and (2) row entities. For example, $A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 3 \\ 1 & 3 & 1 \end{bmatrix}$ with columns $[e_1, e_2, e_3]$, will produce the flattened sequence $[e_1, e_2, e_3, 1, 2, 1, 0, 3, 1]$. IVA injects order-invariance in the representations of entities and relations. For link prediction, order-invariance is useful when aggregating information on a neighborhood.

4.5 Local Clustering Coefficient (LCC)

The clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. For LCC, we use the same inputs as with KHN in section 4.3. LCC is quasi-local graph structural information since we are trying to predict properties of the graph k-hops away. LCC has a similar motivation than KHN and is related to Adamic-Adar and Common Neighbor. However, we are trying to estimate the local clustering coefficient $c_{e_i}$ of a subgraph $k$-hops away from $e_i$. The regression loss is the MSE:

$L_{LCC} = \sum_{i=1}^{k-1} (f(e_i) - c_{e_i})^2.$

In general, nodes with a higher degree tend to have a higher LCC, because they are more likely to have a large number of neighbors. Node degree is used to assess neighborhood quality in link prediction algorithms such as Common Neighbor (Newman 2001). Our ablation study shows that IVA and LCC are useful when combined with other pre-training signals.

5 Baselines

In table 2, we compare our method against strong and recent KG Embedding methods. BOXE (Abboud et al. 2020) is a spatio-translational graph embedding model that uses logical rules (similar to paths) and that can support hierarchical inference patterns and higher-arity relations (knowledge hypergraphs or KHG). This is one of the rare methods that can be applied to both graphs and hypergraphs. BOXE achieves state-of-the-art results on JF17K while remaining competitive on FB15K-237 and WN18RR. It is
unclear however if BOXE is pretrained[4]. To our knowledge, DRGI [Liang et al. 2021] is the only other KG completion method that was pretrained using only signals from the graph structure. Similarly to Deep Graph Info-max [Veličković et al. 2019], DRGI is pretrained on artifacts of the graph structure by maximizing the mutual information between local and global graph representations. We were not able to ascertain at the moment if DRGI is applicable to KHGs.

Both LP-BERT [Li, Yi, and He 2022] and STAR [Wang et al. 2021] are Transformer-based Contextualized KG Embedding methods. The two techniques are improvements over KG-BERT [Yao, Mao, and Luo 2019] that uses textual descriptions of entities and relations on top of a BERT model, which is pretrained on a large text corpus[4]. LP-BERT also uses a multitask pretraining step by jointly training on three denoising tasks: random word masking in the descriptions, entity masking, and relation masking. STAR combines KG-BERT and TransE [Bordes et al. 2013]. STAR contextualizes the translation function by embedding the head and relation descriptions with BERT. The authors claim that the technique is structure-aware since translation-based graph embedding approaches conduct structure learning by measuring spatial distance. Note that KHG tuples in JF17K can have up to 6 entities. In most cases, the description of all entities in a tuple exceeds the standard BERT maximum sequence length of 1024. For this reason, we were not able to apply LP-BERT and STAR to JF17K.

6 Experimental Set-Up

We use FB15K-237 [Bollacker et al. 2008] (extracted from Wikidata, formerly Freebase) and WN18RR [Dettmers et al. 2018]. We also test our method on a hypergraph link prediction task based off the JF17K [Wen et al. 2016] dataset. Note that all datasets are heterogeneous graphs. We see from the Table 3 that we used datasets with a varying number of entities $|\mathcal{E}|$, number of relations $|\mathcal{R}|$, number of examples, density, and arity.

| Dataset | $|\mathcal{E}|$ | $|\mathcal{R}|$ | #train | #valid | #test | density | arity |
|---------|---------------|---------------|--------|--------|-------|---------|-------|
| FB15K-237 | 14,541 | 237 | 272,115 | 17,535 | 20,466 | 18.7 | 2 |
| WN18RR | 40,943 | 11 | 86,835 | 3,034 | 3,134 | 2.1 | 2 |
| JF17K | 29,177 | 327 | 61,911 | 15,822 | 24,915 | 35.9 | $\geq$2 |

Table 3: Dataset Statistics.

The individual pretraining signals seem to lower precision (H@1) and increase recall (H@10) compared to KG-TRSF from scratch. Further, individual pretraining signals have a larger positive effect as graph density increases. Individual pretraining signals typically show MRR improvements for JF17K, the densest of the KGs.

### Training and Evaluation

MRR is the Mean Reciprocal Rank and H@1(3,10) are HIT@ measures all commonly used in link prediction [Mohamed et al. 2019]. For all tasks, we use the same autoregressive Transformer model that applies a transformation $f$ on the input and that is optimized with different loss functions. Our Transformer model is a single monolithic architecture that uses a masking scheme similar to UniLM [Dong et al. 2019], allowing the model to play a variety of roles (encoder-only, decoder-only, or encoder-decoder) and tackle a variety of objectives (classification, regression or sequence generation). In our experiments, we have used $L = 12$ Transformer layers, with $D = 256$ hidden dimension, $A = 12$ self-attention heads, an intermediate layer of size $2D$. For path algorithms, we truncate our path sequences (max hops) to $l = 4$, and, for neighborhood-based algorithms, we limit $k = 3$. The pretraining data generated from each dataset is applied to separate model instances. For each dataset, we save the weights of the pretrained model that performs best on the evaluation set of the link prediction task. During link prediction finetuning, we use a dropout rate $\tau$, a label smoothing rate $\eta$, and a learning rate of $\eta = 5^{-4}$. In our multitask setting (ALL), each task is sampled from the uniform distribution $\frac{1}{|D|}$, for dataset $D_t$ of a pretraining task $t$. A batch may therefore contain multiple tasks. We weight the losses of tasks from $\tau = \{\text{SP, IP, KHN, LCC, IVA}\}$ according the available pretraining dataset size such that $L_{ALL} = \sum_{t \in \tau} \alpha_t L_t$, where the $\alpha_t = |D_t|/\sum_{t \in \tau} |D_t|$. For KHN, note that if $|E_{\mathcal{K} - 1}| > 1024$, we clip the token sequence for a maximum length of 1024. At inference, each entity in $Q_t$ is masked and evaluated once. We use two evaluation metrics: HIT@n and Mean Reciprocal Rank (MRR).

### Experiments

We compare our results with several strong baselines described in Section 5. Our results are presented in Table 2. Our baseline model, KG-TRSF or “CoKe” [Wang et al. 2019], is a Transformer model that is trained from scratch on link prediction. It is important to keep in mind the dataset properties in Table 3 since they explain performance variations of the most recent techniques. For example, BOXE is strongest for higher arity datasets such as JF17K but lags behind on arity-2 KGs. Similarly, STAR has much better scores when evaluated on high density graphs such as FB15K-237, while LP-BERT is best for low density graphs such as WN18RR. KG-TRSF ALL however consistently performs well on three types of graphs.

Our new algorithm IP provides benefit over KG-TRSF across all datasets. Further, we see that all H@ measures and MRR are typically higher for all pretraining signals compared to KG-TRSF. Moreover, all pretraining signals (except LCC) provide gains on the hypergraph JF17K compared to KG-TRSF. We see that using ALL pretraining tasks jointly provides 5% relative MRR score increase over the unpretrained baseline, and often surpassing the competitive performances of state-of-the-art models. In a multitask setting that combines all pretraining tasks, our method surpasses
the latest and strong performing knowledge graph embedding methods on all metrics for FB15K-237, on MRR and Hit@1 for WN18RR and on MRR and hit@10 for JF17K (a knowledge hypergraph dataset). The ALL results are in-line with recent graph pretraining methods (Hu et al. 2020b, Hu et al. 2020c) that show that multitask pretraining performs better than any individual task. Interestingly, our path-based algorithm IP provides the largest single task increase in performance over the unpretrained baseline.

Except for our WN18RR results, path-based pretraining surpasses neighborhood-based pretraining. Compared to SP, we hypothesize that IP provides a higher quality pretraining signal for two reasons: (1) paths are more diverse and (2) paths are semantically closer to a query \( Q_r \). SP produces paths that have often redundant first 2-hop relations on a k-hop path \( \{P_{rel}\}_r \). Further, all the paths have the same minimum length and often transit through high degree entities. As seen in Section 4.1, high degree entities do not necessarily provide the most meaningful reasoning chains; though the IP-based reasoning chains seem more semantically relevant. For example, for a masked query “Spain”, “form of government” and \([\text{mask}]\)”, IP paths are “[military conflict/combattants”, “international organization/member states”], “[adjoining relationship/adjoints”, “continents/countries within”] or “[organization member/member of”, “international organization/member states”].

### 8 Analysis and Ablation

In this section, we discuss the choice and combination of tasks. An ablation study is presented in table 4 that allows us to compare various multitask combinations. We first notice that given SP, IP or SP + IP relational path-based pretraining schemes, adding any other pretraining signal (KHN, LCC or IVA) typically results in an improvement. KHN provides the largest MRR increase when combined with relational path-based signals, with SP + IP + KHN providing a 0.06 improvement in MRR. IVA comes in as the second best signal to combine and LCC only improving MRR slightly.

| Pretraining Signal Combination (MRR)            |   |
|-----------------------------------------------|---|
| SP                                           | 0.368 |
| + KHN                                        | 0.372 |
| + LCC                                        | 0.369 |
| + IVA                                        | 0.370 |
| SP + IP                                      | 0.372 |
| + KHN                                        | 0.375 |
| + LCC                                        | 0.372 |
| + IVA                                        | 0.374 |

Table 4: Task combination ablation study on FB15K-237 using MRR. Adding individual signals to SP, IP or SP + IP pretraining schemes.

### 9 Conclusion

We have investigated the effectiveness of five graph algorithmic pretraining schemes that do not rely on external data sources. On three downstream KG completion tasks, we found that KG-TRSF: (1) multitask pretraining results in performance increases, (2) generally, pretrained models exhibit better improvements in recall (H@10) rather than precision (H@1), and (3) the IP pretraining tasks work best. Our study has important implications since it shows that using various graph structural signals that do not rely on external data can outperform strong baselines pretrained with external data. A deeper study on graph topology and pretraining is still required (e.g., number of unique relation types, graph diameter). In future work, it would be interesting to investigate our single-datasets pretraining with entities and relations encoded with a BERT model.

### Limitations.

Our technique has a few limitations. First, the pretraining signals that we used in this paper require a highly adaptive model. Out of the five pretraining schemes, two include the task of relational path sequence generation (SP and IP) and another is a local clustering coefficient regression task (LCC). Such tasks typically cannot be performed by most Graph Neural Networks or Graph Embedding methods. However, the versatility of our Transformer-based technique also means that our model can be pretrained on multiple modalities (Mustafa et al. 2022, Lee et al. 2022). For example, pretraining with text and KGs has already proven very powerful in language generation tasks (Agarwal et al. 2021). The point of our study was to show that we can surpass other methods such as DRGI and IP-BERT (see Section 5 for an overview of baselines) that use pretrained BERT models and entity descriptions. We suspect that enhancing our entity and relation representations with text will only make the model even stronger at KG completion. Finally, we notice that jointly training on ALL pretraining tasks yields the best results. However, since it is possible that negative task interference occurs (a negative side effect of multitask learning (Zhang, Deng, and Wu 2020, Pilault, hattami, and Pal 2021)), a more throughout study of task combinations can help unlock even better performances for each specific dataset.
References

Abboud, R.; Ceylan, I.; Lukasiewicz, T.; and Salvatori, T. 2020. Boxe: A box embedding model for knowledge base completion. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., Advances in Neural Information Processing Systems, volume 33, 9649–9661. Curran Associates, Inc.

Adamic, L. A., and Adar, E. 2003. Friends and neighbors on the web. Social Networks 25(3):211–230.

Agarwal, O.; Ge, H.; Shakeri, S.; and Al-Rfou, R. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 3554–3565. Online: Association for Computational Linguistics.

Aziz, F.; Gul, H.; Uddin, M. I.; and Gkoutos, G. 2020. Path-based extensions of local link prediction methods for complex networks. Scientific Reports 10.

Bao, H.; Dong, L.; Piao, S.; and Wei, F. 2022. BEit: BERT pre-training of image transformers. In International Conference on Learning Representations.

Benhidour, H.; Almeshkhas, L.; and Kerrache, S. 2022. An approach for link prediction in directed complex networks based on asymmetric similarity-popularity. ArXiv abs/2207.07399.

Bollacker, K. D.; Evans, C.; Paritosh, P. K.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In SIGMOD Conference.

Bordes, A.; Usunier, N.; Garcia-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for multi-relational data. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, 2787–2795. Red Hook, NY, USA: Curran Associates Inc.

Das, R.; Neelakantan, A.; Belanger, D.; and McCallum, A. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, 132–141. Valencia, Spain: Association for Computational Linguistics.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In McIlraith, S. A., and Weinberger, K. Q., eds., Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, 1811–1818. AAAI Press.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H. 2019. Unified language model pre-training for natural language understanding and generation. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d’Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 13042–13054.

Fatemi, B.; Taslakian, P.; Vazquez, D.; and Poole, D. 2020. Knowledge hypergraphs: Prediction beyond binary relations. In Bassiere, C., ed., Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, 2191–2197. International Joint Conferences on Artificial Intelligence Organization. Main track.

Fatemi, B.; El Asri, L.; and Kazemi, S. M. 2021. Slaps: Self-supervision improves structure learning for graph neural networks. Advances in Neural Information Processing Systems 34:22667–22681.

Goldberg, H. G., and Reddy, R. 1977. Speech understanding systems. Summary of results of the five-year research effort at Carnegie-Mellon University. Interim Report Carnegie-Mellon Univ.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 855–864.

Guo, S.; Wang, Q.; Wang, B.; Wang, L.; and Guo, L. 2015. Semantically smooth knowledge graph embedding. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 84–94. Beijing, China: Association for Computational Linguistics.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017a. Inductive representation learning on large graphs. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017b. Representation learning on graphs: Methods and applications. IEEE Data Eng. Bull. 40(3):52–74.

Hasanzadeh, A.; Hajiramezanali, E.; Narayanan, K.; Duffield, N.; Zhou, M.; and Qian, X. 2019. Semi-implicit graph variational auto-encoders. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d’Alché-Buc, F.; Fox, E.; and Garnett, R., eds., Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; and Leskovec, J. 2020a. Strategies for pre-training graph neural networks. In International Conference on Learning Representations.
Hu, W.; Liu*, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; and Leskovec, J. 2020b. Strategies for pre-training graph neural networks. In International Conference on Learning Representations.

Hu, Z.; Dong, Y.; Wang, K.; Chang, K.; and Sun, Y. 2020c. GPT-GNN: generative pre-training of graph neural networks. In Gupta, R.; Liu, Y.; Tang, J.; and Prakash, B. A., eds., KDD ‘20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23–27, 2020, 1857–1867. ACM.

Jiang, X.; Wang, Q.; Qi, B.; Qiu, Y.; Li, P.; and Wang, B. 2017. Attentive path combination for knowledge graph completion. In Zhang, M.-L., and Noh, Y.-K., eds., Proceedings of the Ninth Asian Conference on Machine Learning, volume 77 of Proceedings of Machine Learning Research, 590–605. Yonsei University, Seoul, Republic of Korea: PMLR.

Katz, L. 1953. A new status index derived from sociometric analysis. Psychometrika 18:39–43.

Khasanova, R., and Frossard, P. 2017. Graph-based isometry invariant representation learning. In Precup, D., and Teh, Y. W., eds., Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017, volume 70 of Proceedings of Machine Learning Research, 1847–1856. PMLR.

Kipf, T. N., and Welling, M. 2016. Variational graph auto-encoders. CoRR abs/1611.07308.

Kullback, S., and Leibler, R. A. 1951. On Information and Sufficiency. The Annals of Mathematical Statistics 22(1):79 – 86.

Lee, K.-H.; Nachum, O.; Yang, M.; Lee, L.; Freeman, D.; Xu, W.; Guadarrama, S.; Fischer, I.; Jang, E.; Michalewski, H.; and Mordatch, I. 2022. Multi-game decision transformers.

Li, D.; Yi, M.; and He, Y. 2022. LP-BERT: multi-task pre-training knowledge graph BERT for link prediction. CoRR abs/2201.04843.

Liang, S.; Shao, J.; Zhang, D.; Zhang, J.; and Cui, B. 2021. Drgi: Deep relational graph infomax for knowledge graph completion. IEEE Transactions on Knowledge and Data Engineering 1–1.

Lin, Y.; Liu, Z.; Luan, H.; Sun, M.; Rao, S.; and Liu, S. 2015. Modeling relation paths for representation learning of knowledge bases. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 705–714. Lisbon, Portugal: Association for Computational Linguistics.

Mazumder, S., and Liu, B. 2017. Context-aware path ranking for knowledge base completion. In International Joint Conference on Artificial Intelligence.

Miller, G. A. 1994. WordNet: A lexical database for English. In Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994.

Mohamed, S. K.; Nováček, V.; Vandebussche, P.-Y.; and Muñoz, E. 2019. Loss functions in knowledge graph embedding models. In DL4KG@ESWC.
Trinh, T. H.; Luong, M.; and Le, Q. V. 2019. Selfie: Self-supervised pretraining for image embedding. CoRR abs/1906.02940.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep graph infomax. In International Conference on Learning Representations.

Wang, Z., and Li, J. 2016. Text-enhanced representation learning for knowledge graph. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI’16, 1293–1299. AAAI Press.

Wang, Q.; Huang, P.; Wang, H.; Dai, S.; Jiang, W.; Liu, J.; Lyu, Y.; Zhu, Y.; and Wu, H. 2019. Coke: Contextualized knowledge graph embedding. CoRR abs/1911.02168.

Wang, P.; Peng, N.; Ilievski, F.; Szekely, P.; and Ren, X. 2020. Connecting the dots: A knowledgeable path generator for commonsense question answering. In Findings of the Association for Computational Linguistics: EMNLP 2020, 4129–4140. Online: Association for Computational Linguistics.

Wang, B.; Shen, T.; Long, G.; Zhou, T.; Wang, Y.; and Chang, Y. 2021. Structure-augmented text representation learning for efficient knowledge graph completion. In Proceedings of the Web Conference 2021, WWW ’21, 1737–1748. New York, NY, USA: Association for Computing Machinery.

Wang, Q.; Wang, B.; and Guo, L. 2015. Knowledge base completion using embeddings and rules. In Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI’15, 1859–1865. AAAI Press.

Wang, S.; Zhang, Y.; and Liao, Z. 2019/07. Review on the knowledge graph in robotics domain. In Proceedings of the 3rd International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2019), 424–431. Atlantis Press.

Watts, D., and Strogatz, S. 1998. Collective dynamics of ’small-world’ networks. Nature 1(393):440–442.

Wei, Z.; Zhao, J.; Liu, K.; Qi, Z.; Sun, Z.; and Tian, G. 2015. Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM ’15, 1331–1340. New York, NY, USA: Association for Computing Machinery.

Wen, J.; Li, J.; Mao, Y.; Chen, S.; and Zhang, R. 2016. On the representation and embedding of knowledge bases beyond binary relations. In Kambhampati, S., ed., Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, 1300–1307. IJCAI/AAAI Press.

Xie, R.; Liu, Z.; Jia, J.; Luan, H.; and Sun, M. 2016. Representation learning of knowledge graphs with entity descriptions. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI’16, 2659–2665. AAAI Press.

Yao, L.; Mao, C.; and Luo, Y. 2019. Kg-bert: Bert for knowledge graph completion. ArXiv abs/1909.03193.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do transformers really perform badly for graph representation? In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., Advances in Neural Information Processing Systems.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks?. In Advances in neural information processing systems, 3320–3328.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph contrastive learning with augmentations. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., Advances in Neural Information Processing Systems, volume 33, 5812–5823. Curran Associates, Inc.

You, J.; Ying, R.; and Leskovec, J. 2019. Position-aware graph neural networks. In International conference on machine learning, 7134–7143. PMLR.

Zhang, W.; Deng, L.; and Wu, D. 2020. Overcoming negative transfer: A survey. CoRR abs/2009.00909.