REFACTOR GNNS: Revisiting Factorisation-based Models from a Message-Passing Perspective

Yihong Chen[ϒϑ] **Pushkar Mishra**[ϑ] **Luca Franceschi**[Π*] **Pasquale Minervini**[ϒΩ*]
Pontus Stenetorp[ϒ] **Sebastian Riedel**[ϒϑ]
[ϒ]UCL Centre for Artificial Intelligence, London, United Kingdom
[ϑ]Meta AI, London, United Kingdom
[Π]Amazon Web Services, Berlin, Germany
[Ω]School of Informatics, University of Edinburgh, Edinburgh, United Kingdom
{yihong.chen, p.stenetorp, s.riedel}@cs.ucl.ac.uk
pushkarmishra@meta.com franuluc@amazon.de p.minervini@ed.ac.uk

Abstract

Factorisation-based Models (FMs), such as DistMult, have enjoyed enduring success for Knowledge Graph Completion (KGC) tasks, often outperforming Graph Neural Networks (GNNs). However, unlike GNNs, FMs struggle to incorporate node features and generalise to unseen nodes in inductive settings. Our work bridges the gap between FMs and GNNs by proposing REFACTOR GNNS. This new architecture draws upon *both* modelling paradigms, which previously were largely thought of as disjoint. Concretely, using a message-passing formalism, we show how FMs can be cast as GNNs by reformulating the gradient descent procedure as message-passing operations, which forms the basis of our REFACTOR GNNS. Across a multitude of well-established KGC benchmarks, our REFACTOR GNNS achieve comparable transductive performance to FMs, and state-of-the-art inductive performance while using an order of magnitude fewer parameters.

1 Introduction

In recent years, machine learning on graphs has attracted significant attention due to the abundance of graph-structured data and developments in graph learning algorithms. Graph Neural Networks (GNNs) have shown state-of-the-art performance for many graph-related problems, such as node classification [16] and graph classification [7]. Their main advantage is that they can easily be applied in an inductive setting: generalising to new nodes and graphs without re-training. However, despite many attempts at applying GNNs for multi-relational link prediction tasks such as Knowledge Graph Completion [26], there are still few positive results compared to factorisation-based models (FMs) [45, 39]. As it stands, GNNs either – after resolving reproducibility concerns – deliver significantly lower performance [23, 36] or yield negligible performance gains at the cost of highly sophisticated architecture designs [43]. A notable exception is NBFNet [49], but even here improvements come at the price of high computational inference costs compared to FMs. Furthermore, it is unclear how NBFNet could incorporate node features, which – as we will see in this work – leads to remarkably lower performance in an inductive setting. On the other hand FMs, despite being a simpler architecture, have been found to be very accurate for knowledge graph completion when coupled with appropriate training strategies [30] and training objectives [17, 2]. However, they also come with shortcomings in that they, unlike GNNs, can not be applied in an inductive setting.

[*]Work done while at UCL.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Given the respective strengths and weaknesses of FMs and GNNs, *can we bridge these two seemingly different model categories?* While exploring this question, we make the following contributions:

- By reformulating gradient descent on node embeddings using message-passing primitives, we show a practical connection between FMs and GNNs, in that: FMs can be treated as a special instance of GNNs, but with infinite neighbourhood, layer-wise training and a global normaliser.[2]
- Based on this connection, we propose a new family of architectures, referred to as REFACTOR GNNS, which interpolates between FMs and GNNs. In essence, REFACTOR GNNS inductivise FMs by using a finite number of message-passing layers, and incorporating node features.
- Through an empirical investigation across 15 well-established inductive and transductive benchmarks, we find REFACTOR GNNS achieve state-of-the-art inductive performance and comparable transductive performance to FMs – despite using an order of magnitude fewer parameters.

2 Background

Knowledge Graph Completion (KGC) [25] is a canonical task of multi-relational link prediction. The goal is to predict missing edges given the existing edges in the knowledge graph. Formally, a knowledge graph contains a set of entities (nodes) $\mathcal{E} = \{1, \ldots, |\mathcal{E}|\}$, a set of relations (edge types) $\mathcal{R} = \{1, \ldots, |\mathcal{R}|\}$, and a set of typed edges between the entities $\mathcal{T} = \{(v_i, r_i, w_i)\}_{i=1}^{|\mathcal{T}|}$, where each triplet (v_i, r_i, w_i) indicates a relationship of type $r_i \in \mathcal{R}$ between the *subject* $v_i \in \mathcal{E}$ and the *object* $w_i \in \mathcal{E}$. Given a node v, we denote its *outgoing* 1-hop neighbourhood as the set of relation-object pairs $\mathcal{N}^1_+[\mathrm{v}] = \{(r, o) \mid (\mathrm{v}, r, o) \in \mathcal{T}\}$, its *incoming* 1-hop neighbourhood as the set of subject-relation pairs $\mathcal{N}^1_-[\mathrm{v}] = \{(r, s) \mid (s, r, \mathrm{v}) \in \mathcal{T}\}$, and $\mathcal{N}^1[v] = \mathcal{N}^1_+[v] \cup \mathcal{N}^1_-[v]$ the union of the two. We denote the neighbourhood of v under a specific relation r as $\mathcal{N}^1_\pm[r, v]$. Entities may come with features $X \in \mathbb{R}^{|\mathcal{E}| \times K}$ for describing them, such as textual encodings of their names and/or descriptions. Given a (training) knowledge graph, KGC is evaluated by answering $(v, r, ?)$-style queries i.e. predicting the object given the subject and relation in the triplet.

Multi-relational link prediction models can be trained via maximum likelihood, by fitting a parameterized conditional categorical distribution $P_\theta(w \mid v, r)$ over the candidate objects of a relation, given the subject v and the relation type r:

$$P_\theta(w|\,\mathrm{v}, \mathrm{r}) = \frac{\exp \Gamma_\theta(\mathrm{v}, \mathrm{r}, w)}{\sum_{u \in \mathcal{E}} \exp \Gamma_\theta(\mathrm{v}, \mathrm{r}, u)} = \mathrm{Softmax}(\Gamma_\theta(\mathrm{v}, \mathrm{r}, \cdot))[w]. \tag{1}$$

Here $\Gamma_\theta : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$ is a *scoring function*, which, given a triplet (v, r, w), returns the likelihood that the corresponding edge appears in the knowledge graph. We illustrate our derivations using DistMult [45] as the score function Γ and defer extensions to general score functions, e.g. ComplEx [39], to the appendix. For DistMult, the score function Γ_θ is defined as the tri-linear dot product of the vector representations corresponding to the subject, relation, and object of the triplet:

$$\Gamma_\theta(v, r, w) = \langle f_\phi(v), f_\phi(w), g_\psi(r) \rangle = \sum_{i=1}^{K} f_\phi(v)_i f_\phi(w)_i g_\psi(r)_i, \tag{2}$$

where $f_\phi : \mathcal{E} \to \mathbb{R}^K$ and $g_\psi : \mathcal{R} \to \mathbb{R}^K$ are learnable maps parameterised by ϕ and ψ that encode entities and relation types into K-dimensional vector representations, and $\theta = (\phi, \psi)$. We will refer to f and g as the entity and relation *encoders*, respectively. If we define the data distribution as $P_D(x) = \frac{1}{|\mathcal{T}|} \sum_{(v,r,w) \in \mathcal{T}} \delta_{(v,r,w)}(x)$, where $\delta_{(v,r,w)}(x)$ is a Dirac delta function at (v, r, w), then the objective is to learn the model parameters θ by minimising the expected negative log-likelihood $\mathcal{L}(\theta)$ of the ground-truth entities for the queries $(v, r, ?)$ obtained from $\mathcal{T}$:

$$\arg\min_\theta \mathcal{L}(\theta) \quad \text{where} \quad \mathcal{L}(\theta) = -\mathbb{E}_{x \sim P_D}[\log(P_\theta(w|v, r)] = -\frac{1}{|\mathcal{T}|} \sum_{(v,r,w) \in \mathcal{T}} \log P_\theta(w|v, r). \tag{3}$$

During inference, we use P_θ for determining the plausibility of links not present in the training graph.

[2] The traditional view is that *the transductive nature of FMs stem from their need to retrain on new nodes*, a view which we further underpin by also observing that *FMs are not inductive due to the need for infinite layers of on-the-fly message-passing.*

2.1 Factorisation-based Models for KGC

In factorisation-based models, which we assume to be DistMult, f_ϕ and g_ψ are simply parameterised as look-up tables, associating each entity and relation with a continuous distributed representation:

$$f_\phi(v) = \phi[v],\ \phi \in \mathbb{R}^{|\mathcal{E}|\times K} \quad \text{and} \quad g_\psi(r) = \psi[r],\ \psi \in \mathbb{R}^{|\mathcal{R}|\times K}. \tag{4}$$

2.2 GNN-based Models for KGC

GNNs were originally proposed for node or graph classification tasks [8, 32]. To adapt them to KGC, previous work has explored two different paradigms: *node-wise entity representations* [33] and *pair-wise entity representations* [37, 49]. Though the latter paradigm has shown promising results, it requires computing representations for all pairs of nodes, which can be computationally expensive for large-scale graphs with millions of entities. Additionally, node-wise representations allow for using a single evaluation of $f_\phi(v)$ for multiple queries involving v.

Models based on the first paradigm differ from pure FMs only in the entity encoder and lend themselves well for a fairer comparison with pure FMs. We will therefore focus on this class and leave the investigation of pair-wise representations to future work. Let $q_\phi : \mathcal{G} \times \mathcal{X} \to \bigcup_{S\in\mathbb{N}^+} \mathbb{R}^{S\times K}$ be a GNN encoder, where $\mathcal{G} = \{G \mid G \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$ is the set of all possible multi-relational graphs defined over $\mathcal{E}$ and $\mathcal{R}$, and $\mathcal{X}$ is the input feature space, respectively. Then we can set $f_\phi(v) = q_\phi(\mathcal{T}, X)[v]$. Following the standard message-passing framework [7, 9] used by the GNNs, we view $q_\phi = q^L \circ ... \circ q^1$ as the recursive composition of $L \in \mathbb{N}^+$ layers that compute intermediate representations h^l for $l \in \{1, \dots, L\}$ (and $h^0 = X$) for all entities in the KG. Each layer is made up of the following three functions:

1. A *message function* $q^l_{\text{M}} : \mathbb{R}^K \times \mathcal{R} \times \mathbb{R}^K \to \mathbb{R}^K$ that computes the message along each edge. Given an edge $(v, r, w) \in \mathcal{T}$, q^l_{M} not only makes use of the node states $h^{l-1}[v]$ and $h^{l-1}[w]$ (as in standard GNNs) but also uses the relation r; denote the message as
$$m^l[v, r, w] = q^l_{\text{M}}\left(h^{l-1}[v], r, h^{l-1}[w]\right);$$
2. An *aggregation function* $q^l_{\text{A}} : \bigcup_{S\in\mathbb{N}} \mathbb{R}^{S\times K} \to \mathbb{R}^K$ that aggregates all messages from the 1-hop neighbourhood of a node; denote the aggregated message as
$$z^l[v] = q^l_{\text{A}}\left(\{m^l[v, r, w] \mid (r, w) \in \mathcal{N}^1[v]\}\right);$$
3. An *update function* $q^l_{\text{U}} : \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}^K$ that produces the new node states h^l by combining previous node states h^{l-1} and the aggregated messages z^l:
$$h^l[v] = q^l_{\text{U}}(h^{l-1}[v], z^l[v]).$$

Different parameterisations of q^l_{M}, q^l_{A}, and q^l_{U} lead to different GNNs. For example, R-GCNs [33] define the q^l_{M} function using per-relation the linear transformations $m^l[v, r, w] = \frac{1}{\mathcal{N}^1[r,v]} W^l_r h^{l-1}[w]$; q^l_{A} is implemented by a summation and q^l_{U} is a non-linear transformation $h^l[v] = \sigma(z^l[v] + W^l_0 h^{l-1}[v])$, where σ is the sigmoid function. For each layer, the learnable parameters are $\{W^l_r\}_{r\in\mathcal{R}}$ and W^l_0, all of which are matrices in $\mathbb{R}^{K\times K}$.

3 Implicit Message-Passing in FMs

The sharp difference in analytical forms might give rise to the misconception that GNNs incorporate message-passing over the neighbourhood of each node (up to L-hops), while FMs do not. In this work, we show that by explicitly considering the training dynamics of FMs, we can uncover and analyse the hidden message-passing mechanism within FMs. In turn, this will lead us to the formulation of a novel class of GNNs well suited for multi-relational link prediction tasks (Section 4). Specifically, we propose to interpret the FMs' optimisation process of their objective (3) as the entity encoder. If we consider, for simplicity, a gradient descent training dynamic, then

$$f_{\phi^t}(v) = \phi^t[v] = \text{GD}^t(\phi^{t-1}, \mathcal{T})[v] = \underbrace{\text{GD}^t \circ ... \text{GD}^1}_{t}(\phi^0, \mathcal{T})[v], \tag{5}$$

where ϕ^t is the embedding vector at the t-th step, $t \in \mathbb{N}^+$ is the total number of iterations and ϕ^0 is a random initialisation. GD is the gradient descent operator:

$$\mathrm{GD}(\phi, \mathcal{T}) = \phi - \alpha \nabla_\phi \mathcal{L} = \phi + \alpha \sum_{(v,r,w)\in\mathcal{T}} \frac{\partial \log P(w|v,r)}{\partial \phi}, \tag{6}$$

where $\alpha = \beta\,|\mathcal{T}|^{-1}$, with a $\eta > 0$ learning rate. We now dissect Equation (6) in two different (but equivalent) ways. In the first, which we dub the *edge view*, we separately consider each addend of the gradient $\nabla_\phi \mathcal{L}$. In the second, we aggregate the contributions from all the triplets to the update of a particular node. With this latter decomposition, which we call the *node view*, we can explicate the message-passing mechanism at the core of the FMs. While the edge view suits a vectorised implementation better, the node view further exposes the information flow among nodes, allowing us to draw an analogy to message-passing GNNs.

3.1 The Edge View

Each addend of Equation (6) corresponds to a single edge $(v, r, w) \in \mathcal{T}$ and contributes to the update of the representation of all nodes. The update on the representation of the subject $\phi[v]$ is:

$$\mathrm{GD}(\phi, \{(v,r,w)\})[v] = \phi[v] + \alpha \left(\underbrace{g(r) \odot \phi[w]}_{w\to v} \underbrace{- \sum_{u\in\mathcal{E}} P_\theta(u|v,r) g(r) \odot \phi[u]}_{u\to v} \right).$$

The $w \to v$ term indicates information flow from w (a neighbour of v) to v, increasing the score of the gold triplet (v, r, w). The $u \to v$ term indicates information flow from global nodes, decreasing the scores of triplets $(v, r, ?)$ with v as the subject and r as the predicate. Similarly, for the object w,

$$\mathrm{GD}(\phi, \{(v,r,w)\})[w] = \phi[w] + \alpha \underbrace{(1 - P_\theta(w|v,r))\, g(r) \odot \phi[v]}_{v\to w},$$

where, again, the $v \to w$ term indicates information flow from the neighbouring node v. Finally, for the nodes other than v and w, we have

$$\mathrm{GD}(\phi, \{(v,r,w)\})[u] = \phi[u] + \alpha \left(\underbrace{-P_\theta(u|v,r)\phi[v] \odot g(r)}_{v\to u} \right).$$

3.2 The Node View

To fully uncover the message-passing mechanism of FMs, we now focus on the gradient descent operation over a single node $v \in \mathcal{E}$, referred to as the *central node* in the GNN literature. Recalling Equation (6), we have:

$$\mathrm{GD}(\phi, \mathcal{T})[v] = \phi[v] + \alpha \sum_{(v,r,w)\in\mathcal{T}} \frac{\partial \log P(\bar{\mathrm{w}} \,|\, \bar{\mathrm{v}}, \bar{\mathrm{r}})}{\partial \phi[v]}, \tag{7}$$

which aggregates the information stemming from the updates presented in the edge view. The next theorem describes how this total information flow to a particular node can be recast as an instance of message passing (cf. Section 2.2). We defer the proof to the appendix.

Theorem 3.1 (Message passing in FMs)**.** *The gradient descent operator* GD *(Equation* (7)*) on the node embeddings of a DistMult model (Equation* (4)*) with the maximum likelihood objective in Equation* (3) *and a multi-relational graph $\mathcal{T}$ defined over entities $\mathcal{E}$ induces a message-passing operator whose composing functions are:*

$$q_{\mathrm{M}}(\phi[v], r, \phi[w]) = \begin{cases} \phi[w] \odot g(r) & \text{if } (r,w) \in \mathcal{N}_+^1[v], \\ (1 - P_\theta(v|w,r))\phi[w] \odot g(r) & \text{if } (r,w) \in \mathcal{N}_-^1[v]; \end{cases} \tag{8}$$

$$q_{\mathrm{A}}(\{m[v,r,w] \,:\, (r,w) \in \mathcal{N}^1[v]\}) = \sum_{(r,w)\in\mathcal{N}^1[v]} m[v,r,w]; \tag{9}$$

$$q_{\mathrm{U}}(\phi[v], z[v]) = \phi[v] + \alpha z[v] - \beta n[v], \tag{10}$$

where, defining the sets of triplets $\mathcal{T}^{-v} = \{(s, r, o) \in \mathcal{T} \ : \ s \neq v \wedge o \neq v\}$,

$$n[v] = \frac{|\mathcal{N}_+^1[v]|}{|\mathcal{T}|} \mathbb{E}_{P_{\mathcal{N}_+^1[v]}} \mathbb{E}_{u \sim P_\theta(\cdot|v,r)} g(r) \odot \phi[u] + \frac{|\mathcal{T}^{-v}|}{|\mathcal{T}|} \mathbb{E}_{P_{\mathcal{T}^{-v}}} P_\theta(v|s, r) g(r) \odot \phi[s], \tag{11}$$

where $P_{\mathcal{N}_+^1[v]}$ *and* $P_{\mathcal{T}^{-v}}$ *are the empirical probability distributions associated to the respective sets.*

What emerges from the equations is that each GD step contains an explicit information flow from the neighbourhood of each node, which is then aggregated with a simple summation. Through this direct information path, t GD steps cover the t-hop neighbourhood of v. As t goes towards infinity – or in practice – as training converges, FMs capture the global graph structure. The update function (10) somewhat deviates from classic message passing frameworks as $n[v]$ of Equation (11) involves global information. However, we note that we can interpret this mechanism under the framework of augmented message passing [41] and, in particular, as an instance of *graph rewiring*.

Based on Theorem 3.1 and Equation (5), we can now view ϕ as the transient node states h (cf. Section 2.2) and GD on node embeddings as a message-passing layer. This dualism sits at the core of the ReFactor GNN model, which we describe next.

4 REFACTOR GNNs

FMs are trained by minimising the objective (3), initialising both sets of parameters (ϕ and ψ) and performing GD until approximate convergence (or until early stopping terminates the training). The implications are twofold: i) the initial value of the entity lookup table ϕ does not play any major role in the final model after convergence, and ii) if we introduce a new set of entities, the conventional wisdom is to retrain[3] the model on the expanded knowledge graph. This is computationally rather expensive compared to the "inductive" models that require no additional training and can leverage node features like entity descriptions. However, as we have just seen in Theorem 3.1, the training procedure of FMs may be naturally recast as a message-passing operation, which suggests that it is possible to use FMs for inductive learning tasks. In fact, we envision that there is an entire novel spectrum of model architectures interpolating between pure FMs and (various instantiations of) GNNs. Here we propose one simple implementation of such an architecture which we dub REFACTOR GNNs. Figure 1 gives an overview of REFACTOR GNNs.

The REFACTOR Layer A REFACTOR GNN contains L REFACTOR layers, that we derive from Theorem 3.1. Aligning with the notations in Section 2.2, given a KG $\mathcal{T}$ and entity representations $h^{l-1} \in \mathbb{R}^{|\mathcal{E}| \times K}$, the REFACTOR layer computes the representation of a node v as follows:

$$h^l[v] = q^l(\mathcal{T}, h^{l-1})[v] = h^{l-1}[v] - \beta n^l[v] + \alpha \sum_{(r,w) \in \mathcal{N}^1[v]} q_M^l(h^{l-1}[v], r, h^{l-1}[w]), \tag{12}$$

where the terms n^l and q_{M}^l are derived from Equation (11) and Equation (8), respectively. We note that REFACTOR GNNs treat incoming and outgoing neighbourhoods differently instead of treating them equally as in the R-GCN, the first GNN on multi-relational graphs [33].

Equation (12) describes the full batch setting, which can be expensive if the KG contains many edges. Therefore, in practice, whenever the graph is big, we adopt a stochastic evaluation of the REFACTOR layer by decomposing the evaluation into several mini-batches. We partition $\mathcal{T}$ into a set of computationally tractable mini-batches. For each of them, we restrict the neighbourhoods to the subgraph induced by it and readjust the computation of $n^l[v]$ to include only entities and edges present in it. We leave the investigation of other stochastic strategies (e.g. by taking Monte Carlo estimations of the expectations in Equation (11)) to future work. Finally, we cascade the mini-batch evaluation to produce one full layer evaluation.

Training The learnable parameters of REFACTOR GNNs are the relation embeddings ψ, which parameterise $g(r)$ in $q_M^l, l \in [1, L]$. Inspired by [5, 46], we learn ψ by layer-wise (stochastic) gradient descent. This is in contrast to conventional GNN training, where we need to backpropagate through

[3] Typically until convergence, possibly by partially warm-starting θ.

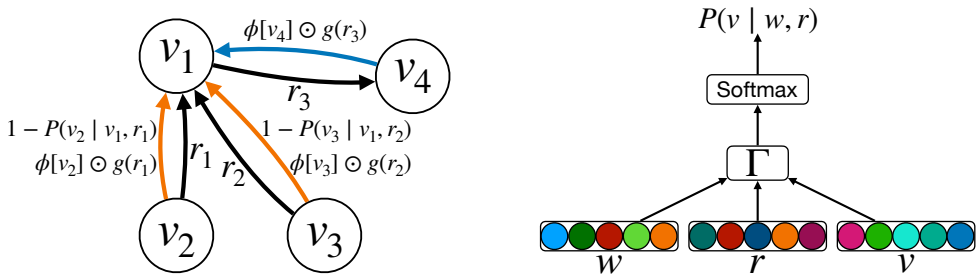

Figure 1: ReFactor GNN architecture – the left figure describes the messages (coloured edges) used to update the representation of node v_1, which depend on the type of relationship between the sender nodes and v_1 in the graph $G = \{(v_2, r_1, v_1), (v_3, r_2, v_1), (v_1, r_3, v_4)\}$; the right figure describes the computation graph for calculating $P(v \mid w, r)$, where $v, w \in \mathcal{E}$ and $r \in \mathcal{R}$: the embedding representations of w, r, and v are used to score the edge (w, r, v) via the scoring function Γ, which is then normalised via the Softmax function.

all the layers. A (full-batch) GD training dynamic for ψ can be written as $\psi_{t+1} = \psi_t - \eta \nabla \mathcal{L}_t(\psi_t)$, where $\mathcal{L}_t(\psi_t) = -|\mathcal{T}|^{-1} \sum_{\mathcal{T}} \log P_{\psi_t}(w|v, r)$, with:

$$P_{\psi_t}(w|v, r) = \text{Softmax}(\Gamma(v, r, \cdot))[w], \qquad \Gamma(v, r, w) = \langle h^t[v], h^t[w], g_{\psi_t}(r) \rangle$$

and the node state update as

$$h^t = \begin{cases} X & \text{if } t \bmod L = 0 \\ q^{t \bmod L}(\mathcal{T}, h^{t-1}) & \text{otherwise.} \end{cases} \tag{13}$$

Implementation-wise, such a training dynamic equals to using an external memory for storing historical node states h^{t-1} akin to the procedure introduced in [5]. The cached node states can then be queried to compute h^t using Equation (12). From this perspective, we periodically clear *the node state cache* every L full batches to force the model to predict based on on-the-fly L-layer message-passing. After training, we obtain ψ^* and perform inference by running L-layer message-passing with ψ^*. In general, L determines the number of effective message-passing layers in REFACTOR GNNS. A larger L enables REFACTOR GNNS to fuse information from more hops of neighbourhoods into the final node representations. In the meantime, it reduces the inductive applicability of REFACTOR GNNS due to over-smoothing and computational requirements. In the extreme case of $L = \infty$, *where we never clear the node state cache during training*, the final cached node states will be used for inference. Note that this latter inference regime is inherently transductive since there will be no cached states for new nodes.

5 Experiments

We perform experiments [4] to answer the following questions regarding REFACTOR GNNS:

- **Q1.** REFACTOR GNNS are derived from a message-passing reformulation of FMs: do they also inherit FMs' predictive accuracy in *transductive* KGC tasks? (Section 5.1)
- **Q2.** REFACTOR GNNS "inductivise" FMs. Are they more statistically accurate than other GNN baselines in *inductive* KGC tasks? (Section 5.2)
- **Q3.** The term $n[v]$ involves nodes that are not in the 1-hop neighbourhood. Is such *augmented message passing* [41] necessary for good KGC performance? (Section 5.3)

For transductive experiments, we used three well-established KGC datasets: *UMLS* [14], *CoDEx-S* [31], and *FB15K237* [38]. For inductive experiments, we used the inductive KGC benchmarks introduced by GraIL [37], which include 12 pairs of knowledge graphs: (*FB15K237_vi*, *FB15K237_vi_ind*), (*WN18RR_vi*, *WN18RR_vi_ind*), and (*NELL_vi*, *NELL_vi_ind*), where $i \in [1, 2, 3, 4]$, and (*_vi*, *_vi_ind*) represents a pair of graphs with a shared relation vocabulary and non-overlapping entities. We follow the standard KGC evaluation protocol by fully ranking all the

[4] The implementation will be available https://github.com/yihong-chen/ReFactorGNN.

Entity Encoder	UMLS	CoDEx-S	FB15K237
R-GCN	-	0.33	0.25
Lookup (FM, specif. DistMult)	0.90	0.43	0.30
REFACTOR GNNS ($L = \infty$)	0.93	0.44	0.33

Table 1: Test MRR for transductive KGC tasks.

candidate entities and computing two metrics using the ranks of the ground-truth entities: Mean Reciprocal Ranking (MRR) and Hit Ratios at Top K (Hits@K) with $K \in [1, 3, 10]$. For the inductive KGC, we additionally consider the partial-ranking evaluation protocol used by GraIL for a fair comparison. Empirically, we find full ranking more difficult than partial ranking, and thus more suitable for reflecting the differences among models on GraIL datasets – we would like to call for future work on GraIL datasets to also adopt a full ranking protocol on these datasets.

Our *transductive* experiments used $L = \infty$, i.e. node states cache is never cleared, as we wanted to see if REFACTOR GNNS ($L = \infty$) can reach the performance of the FMs; on the other hand, in our *inductive* experiments, we used REFACTOR GNNS with $L \in \{1, 2, 3, 6, 9\}$, since we wanted to test their performances in inductive settings akin to standard GNNs. We used a hidden size of 768 for the node representations. All the models are trained using $[128, 512]$ in-batch negative samples and one global negative node for each positive link. We performed a grid search over the other hyper-parameters and selected the best configuration based on the validation MRR. Since training deep GNNs with full-graph message passing might be slow for large knowledge graphs, we follow the literature [10, 50, 47] to sample sub-graphs for training GNNs. Considering that sampling on the fly often prevents high utilisation of GPUs, we resort to a two-stage process: we first sampled and serialised sub-graphs around the target edges in the mini-batches; we then trained the GNNs with the serialised sub-graphs. To ensure we have sufficient sub-graphs for training the models, we sampled for 20 epochs for each knowledge graph, i.e. 20 full passes over the full graph. The sub-graph sampler we currently used is LADIES [50].

5.1 REFACTOR GNNS for Transductive Learning (Q1)

REFACTOR GNNS are derived from the message-passing reformulation of FMs. We expect them to approximate the performance of FMs for transductive KGC tasks. To verify this, we run experiments on the datasets UMLS, CoDEx-S, and FB15K237. For a fair comparison, we use Equation (2) as the decoder and consider i) lookup embedding table as the entity encoder, which forms the FM when combined with the decoder (Section 2.1), and ii) REFACTOR GNNS as the entity encoder. REFACTOR GNNS are trained with $L = \infty$, i.e. we never clear the node state cache. Since transductive KGC tasks do not involve new entities, the node state cache in REFACTOR GNNS can be directly used for link prediction. Table 1 summarises the result. We observe that REFACTOR GNNS achieve a similar or slightly better performance compared to the FM. This shows that REFACTOR GNNS are able to capture the essence of FMs and thus maintain strong at transductive KGC.

5.2 REFACTOR GNNS for Inductive Learning (Q2)

Despite FMs' good empirical performance on transductive KGC tasks, they fail to be inductive as GNNs. According to our reformulation, this is due to the infinite message-passing layers hidden in FMs' optimisation. Discarding infinite message-passing layers, REFACTOR GNNS enable FMs to perform inductive reasoning tasks by learning to use a finite set of message-passing layers for prediction similarly to GNNs.

Here we present experiments to verify REFACTOR GNNS's capability for inductive reasoning. Specifically, we study the task of inductive KGC and investigate whether REFACTOR GNNS can generalise to unseen entities. Following [37], on GraIL datasets, we trained models on the original graph, and run 0-shot link prediction on the *_ind* test graph. Similar to the transductive experiments, we use Equation (2) as the decoder and vary the entity encoder. We denote three-layer REFACTOR GNNS as `REFACTOR GNNS (3)` and six-layer REFACTOR GNNS as `REFACTOR GNNS (6)`. We consider several baseline entity encoders: i) `no-pretrain`, models without any pretraining on the original graph; ii) `GAT(3)`, three-layer graph attention network [40]; iii) *GAT(6)*, six-layer graph attention network; iv) `GraIL`, a sub-graph-based relational GNN [37]; v) `NBFNet`, a path-

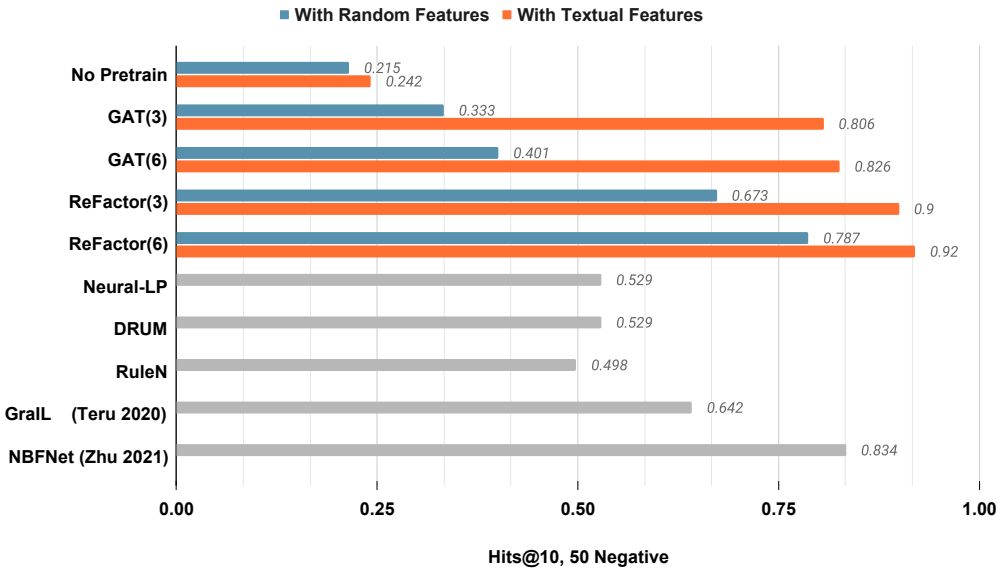

Figure 2: Inductive KGC Performance. Models are trained on the KG *FB15K237_v1* and tested on another KG *FB15K237_v1_ind*, where the entities are completely new. The results of GraIL and NBFNet are taken from [49]. The grey bars indicate methods that are not devised to incorporate node features.

based GNN [49], current SoTA on GraIL datasets. In addition to randomly initialised vectors as the node features, we also used textual node features, RoBERTa [20] Encodings of the entity descriptions, which are produced by SentenceBERT [29]. Due to space reason, we present the results on (*FB15K237_v*1, *FB15K237_v*1*_ind*) in Figure 2. Results on other datasets are similar and can be found in the appendix. We can see that without textual node features, REFACTOR GNNS perform better than GraIL (+23%); with textual node features, REFACTOR GNNS outperform both GraIL (+43%) and NBFNet (+10%), achieving new SoTA results on inductive KGC.

Performance vs Parameter Efficiency as #Message-Passing Layers Increases Usually, as the number of message-passing layers increases in GNNs, the over-smoothing issue occurs while the computational cost also increases exponentially. REFACTOR GNNS avoid this by layer-wise training and sharing the weights across layers. Here we compare REFACTOR GNNS with $\{1, 3, 6, 9\}$ message-passing layer(s) with same-depth GATs – results are summarised in Figure 3. We observe that increasing the number of message-passing layers in GATs does not necessarily improve the predictive accuracy – the best results were obtained with 3 message-passing layers on *FB15K237_v1* while using 6 and 9 layers leads to performance degradation. On the other hand, REFACTOR GNNS obtain consistent improvements when increasing #Layers from 1 to 3, 6, and 9. REFACTOR GNNS $(6, 6)$ and $(9, 9)$ clearly outperform their GAT counterparts. Most importantly, REFACTOR GNNS are more parameter-efficient than GATs, with a constant #Parameters as #Layers increases.

5.3 Beyond Message-Passing (Q3)

As shown by Theorem 3.1, REFACTOR GNNS contain not only terms capturing information flow from the 1-hop neighbourhood, which falls into the classic message-passing framework, but also a term $n[v]$ that involve nodes outside the 1-hop neighbourhood. The term $n[v]$ can be treated as *augmented message-passing* on a dynamically rewired graph [41]. Here we perform ablation experiments to measure the impact of the $n[v]$ term. Table 2 summarises the ablation results: we can see that, without the term $n[v]$, REFACTOR GNNS with random vectors as node features yield a 2% lower MRR, while REFACTOR GNNS with RoBERTa textual encodings as node features produce a 7% lower MRR. This suggests that augmented message-passing also plays a significant role in

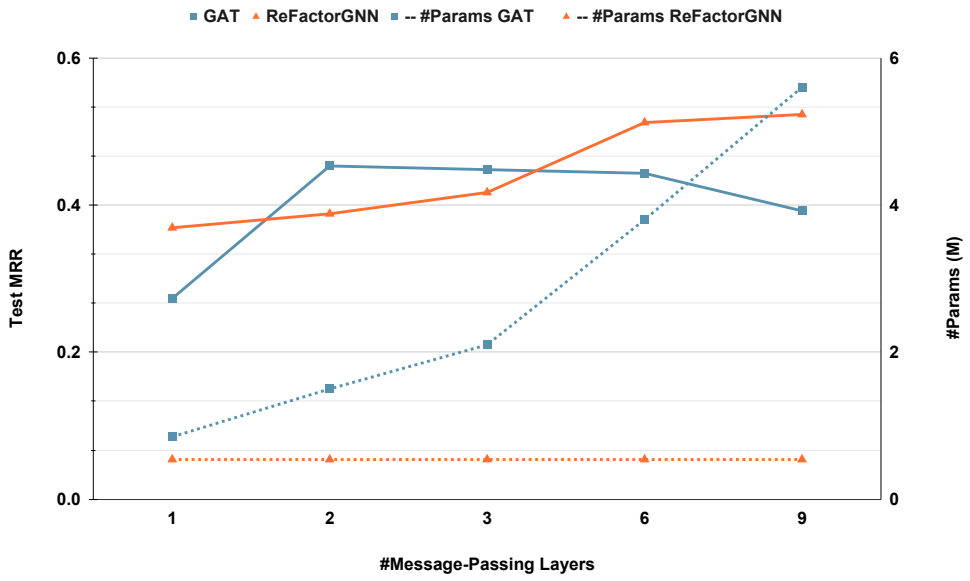

Figure 3: Performance vs Parameter Efficiency as #Layers Increases on *FB15K237_v1*. The left axis is Test MRR while the right axis is #Parameters. The solid lines and dashed lines indicate the changes of Test MRR and the changes of #Parameters.

Test MRR	With Random Features	With Textual Features
with $n[v]$	0.425	0.486
without $n[v]$	0.418	0.452

Table 2: Ablation on $n[v]$ for REFACTOR GNNS (6) trained on *FB15K237_v1*.

REFACTOR GNNS' generalisation properties in downstream link prediction tasks. Future work might gain more insights by further dissecting the $n[v]$ term.

6 Related Work

Multi-Relational Graph Representation Learning Multi-relational graph representation learning concerns graphs with various edge types. Another relevant line of work would be representation learning over heterogeneous graphs, where node types are also considered. Previous work on multi-relational graph representation learning focused either on FMs [27, 39, 45, 17, 26, 3, 24, 2] or on GNN-based models [33, 44, 48, 18]. Similar to a recent finding in a benchmark study over heterogeneous GNNs [21], where the best choices of GNNs for heterogeneous graphs seem to regress to simple homogeneous GNN baselines, the progress of multi-relational graph representation learning also mingles with FMs, the very classic multi-relational link predictors. Recently, FMs were found to be significantly more accurate than GNNs in KGC tasks, when coupled with specific training strategies [30, 12, 17]. While more advanced GNNs [49] for KBC are showing promise at the cost of extra algorithm complexity, little effort has been devoted to establishing the links between plain GNNs and FMs, which are strong multi-relational link predictors despite their simplicity. Our work aims to *align* GNNs with FMs so that we can combine the strengths from both families of models.

Relationships between FMs and GNNs A very recent work [35] builds a theoretical link between structural GNNs and node (positional) embeddings. However, on one end of the link, the second model category encompasses not merely factorisation-based models but also many practical graph neural networks, between which the connection is unknown. Our work instead offers a more practical link between positional node embeddings produced by FMs and positional node embeddings produced by

GNNs, while at the same time focusing on KGC. Beyond FMs in KGC, using graph signal processing theory, [34] show that matrix factorisation (MF) based recommender models correspond to ideal low-pass graph convolutional filters. They also find infinite neighbourhood coverage in MF although using a completely different approach and focusing on a different domain in contrast to our work.

Message Passing Message-passing is itself a broad terminology – people generally talk about it under two different contexts. Firstly, as a computational technique, message passing allows recursively decomposing a global function into simple local, parallelisable computations [22], thus being widely used for solving inference problems in a graphical model. Specifically, we note that message passing-based inference techniques were proposed for matrix completion-based recommendation [15] and Bayesian Boolean data decomposition [28] in the pre-deep-learning era. Secondly, as a paradigm of parameterising learnable functions over *graph-structured data*, message-passing has recently been used to provide a unified reformulation [7] for various GNN architectures, including Graph Attention Networks [40], Gated Graph Neural Networks [19], and Graph Convolutional Networks [16]. In this work, we show that FMs can also be cast as a special type of message-passing GNNs by considering the gradient descent updates [1] over node embeddings as message-passing operations between nodes. To the best of our knowledge, our work is the first to provide such connections between FMs and message-passing GNNs. We show that FMs can be seen as instances of GNNs, with a characteristic feature about the nodes being considered during the message-passing process: our REFACTOR GNNS can be seen as using an *Augmented Message-Passing* process on a dynamically re-wired graph [41].

7 Conclusion & Future Work

Conclusion. Motivated by the need of understanding FMs and GNNs despite the sharp differences in their analytical forms, our work establishes a link between FMs and GNNs on the task of multi-relational link prediction. The reformulation of FMs as GNNs addresses the question why FMs are stronger multi-relational link predictors compared to plain GNNs. Guided by the reformulation, we further propose a new variant of GNNs, REFACTOR GNNS, which combines the strengths of both GNNs and classic FMs. Empirical experiments show that REFACTOR GNNS produce significantly more accurate results than our GNN baselines on inductive link prediction tasks.

Limitations. Since we adopted a two-stage (sub-graph serialisation and then model training) approach instead of online sampling, there can be side effects from the low sub-graph diversity. In our experiments, we used LADIES [50] for sub-graph sampling. Experiments with different sub-graph sampling algorithms, such as GraphSaint [47] might affect the downstream link prediction results. Furthermore, it would be interesting to analyse decoders other than DistMult, as well as additional optimisation schemes beyond SGD and AdaGrad. We do not dive deeper into the expressiveness of REFACTOR GNNS. Nevertheless, we offer a brief discussion in Appendix F.

Future Work. The most direct future work would be using the insight to develop more sophisticated models at the crossroad between FMs and GNNs, e.g. by further parameterising the message/update function. One implication from our work is that reformulating FMs as message-passing enables the idea of "learning to factorize". This might broaden the usages of FMs, going beyond link prediction, to tasks such as graph classification. Another implication comes from our approach of unpacking embedding updates into a series of message-passing operations. This approach can be generalised to other dot-product-based models that use embedding layers for processing the inputs, e.g. transformers. In general, the direction of organising FMs, GNNs, and transformers under the same framework will allow a better understanding of all three models.

Acknowledgements

We would like to thank all reviewers for their constructive comments. We thank people with whom we had some really interesting chats throughout the project: Zhaocheng Zhu, Floris Geerts, Muhan Zhang, Patrick Lewis, Hanqing Zeng, Shalini Maiti, Yifei Shen, Yinglong Xia, Matthias Fey, Jing Zhu, Marc Deisenroth and Maximilian Nickel. Yihong would like to thank the UCL-Facebook AI Research joint PhD program for generously funding her PhD. Pasquale and Pontus were partially funded by the European Union's Horizon 2020 research and innovation programme under grant agreement no. 875160, and by an industry grant from Cisco.

References

[1] Léon Bottou. *Stochastic Gradient Descent Tricks*, pages 421–436. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[2] Yihong Chen, Pasquale Minervini, Sebastian Riedel, and Pontus Stenetorp. Relation prediction as an auxiliary training objective for improving multi-relational graph representations. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

[3] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818. AAAI Press, 2018.

[4] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[5] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Jure Leskovec. Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings. In *ICML*, 2021.

[6] Floris Geerts and Juan L Reutter. Expressiveness and approximation properties of graph neural networks. In *International Conference on Learning Representations*, 2021.

[7] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[8] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 2:729–734 vol. 2, 2005.

[9] William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159.

[10] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[11] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020.

[12] Prachi Jain, Sushant Rathi, Mausam, and Soumen Chakrabarti. Knowledge base completion: Baseline strikes back (again). *CoRR*, abs/2005.00804, 2020.

[13] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. *Advances in neural information processing systems*, 31, 2018.

[14] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, pages 381–388. AAAI Press, 2006.

[15] Byung-Hak Kim, Arvind Yedla, and Henry D Pfister. Imp: A message-passing algorithm for matrix completion. In *2010 6th International Symposium on Turbo Codes & Iterative Information Processing*, pages 462–466. IEEE, 2010.

[16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[17] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 2869–2878. PMLR, 2018.

[18] Ren Li, Yanan Cao, Qiannan Zhu, Guanqun Bi, Fang Fang, Yi Liu, and Qian Li. How does knowledge graph embedding extrapolate to unseen data: a semantic evidence view. *CoRR*, abs/2109.11800, 2021.

[19] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In *ICLR (Poster)*, 2016.

[20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[21] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '21, page 1150–1160, New York, NY, USA, 2021. Association for Computing Machinery.

[22] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[23] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723, 2019.

[24] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Q. Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL-HLT (2)*, pages 327–333. Association for Computational Linguistics, 2018.

[25] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proc. IEEE*, 104(1):11–33, 2016.

[26] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. In *AAAI*, pages 1955–1961. AAAI Press, 2016.

[27] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, pages 809–816. Omnipress, 2011.

[28] Siamak Ravanbakhsh, Barnabás Póczos, and Russell Greiner. Boolean matrix factorization and noisy completion via message passing. In *International Conference on Machine Learning*, pages 945–954. PMLR, 2016.

[29] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[30] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2020.

[31] Tara Safavi and Danai Koutra. Codex: A comprehensive knowledge graph completion benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8328–8350, 2020.

[32] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

[33] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[34] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, Khaled B Letaief, and Dongsheng Li. How powerful is graph convolution for recommendation? *arXiv preprint arXiv:2108.07567*, 2021.

[35] Balasubramaniam Srinivasan and Bruno Ribeiro. On the equivalence between positional node embeddings and structural graph representations. In *International Conference on Learning Representations*, 2020.

[36] Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. A re-evaluation of knowledge graph completion methods. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5516–5522, Online, July 2020. Association for Computational Linguistics.

[37] Komal K. Teru, Etienne G. Denis, and William L. Hamilton. Inductive relation prediction by subgraph reasoning. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 9448–9457. PMLR, 2020.

[38] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66, 2015.

[39] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2071–2080, New York, New York, USA, 20–22 Jun 2016. PMLR.

[40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[41] Petar Veličković. Message passing all the way up, 2022.

[42] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*. OpenReview.net, 2019.

[43] Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. In *International Conference on Learning Representations*, 2020.

[44] Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. In *ICLR*. OpenReview.net, 2020.

[45] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR (Poster)*, 2015.

[46] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. L2-gcn: Layer-wise and learned efficient training of graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2127–2135, 2020.

[47] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

[48] Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhi-Ping Shi, Hui Xiong, and Qing He. Relational graph neural network with hierarchical attention for knowledge graph completion. In *AAAI*, pages 9612–9619. AAAI Press, 2020.

[49] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal A. C. Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *CoRR*, abs/2106.06935, 2021.

[50] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Few-shot representation learning for out-of-vocabulary words. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2019.