Inductive Relation Prediction by Subgraph Reasoning

Komal K. Teru

Computer Science
McGill University, Montreal

August 20, 2020

Acknowledgements

I am deeply grateful to my supervisor William Hamilton for his invaluable supervision on my research and constant mentorship throughout my master's program. It would have been impossible to finish the thesis without Will's guidance and encouragement. I also extend my thanks to Mark Coates for reviewing and providing detailed feedback on the initial drafts of this thesis.

I thank Etienne Denis for his help in getting the experiment pipeline of GraIL off the ground. I extend my gratitude to lab members Jin Dong, Koustuv Sinha, and Devendra Singh for their support and discussions on my project when it was in a rough patch.

Finally, I am grateful to my parents, back in India, for giving me the opportunity to pursue my master's and to my partner and most trusted friend, Alekhya, for providing me with unconditional support when I needed it the most.

Abstract

The dominant paradigm for relation prediction in knowledge graphs involves learning and operating on latent representations (i.e., embeddings) of entities and relations. These methods exploit the local connectivity patterns and homophily in the knowledge graph to make predictions. However, these embedding-based methods do not explicitly capture the compositional logical rules underlying the knowledge graph, and they are limited to the transductive setting, where the full set of entities must be known during training. Here, we propose a graph neural network based relation prediction framework, GraIL, that reasons over local subgraph structures and has a strong inductive bias to learn entity-independent relational semantics. Unlike embedding-based models, GraIL is naturally inductive and can generalize to unseen entities and graphs after training. We provide theoretical proof and strong empirical evidence that GraIL can represent a useful subset of first-order logic and show that GraIL outperforms existing rule-induction baselines in the inductive setting. We also demonstrate significant gains obtained by ensembling GraIL with various knowledge graph embedding methods in the transductive setting, highlighting the complementary inductive bias of our method.

Résumé

Le paradigme dominant pour la prédiction des relations dans les graphes de connaissances implique d'apprendre et d'opérer sur des représentations latentes (c'est-à-dire des plongements) d'entités et de relations. Ces méthodes exploitent les modèles de connectivité locale et l'homophilie dans le graphe des connaissances pour faire des prédictions. Cependant, ces méthodes basées sur l'incorporation ne capturent pas explicitement les règles logiques de composition sous-jacentes au graphe de connaissances, et elles sont limitées au paramètre transducteur, où l'ensemble complet des entités doit être connu pendant la formation. Ici, nous proposons un cadre de prédiction de relations basé sur un réseau de neurones, GraIL, qui raisonne sur les structures locales de sous-graphes et a un fort biais inductif pour apprendre la sémantique relationnelle indépendante de l'entité. Contrairement aux modèles basés sur des plongements, GraIL est naturellement inductif et peut se généraliser à des entités et des graphiques invisibles après la formation. Nous fournissons des preuves théoriques et des preuves empiriques solides que GraIL peut représenter un sous-ensemble utile de logique de premier ordre et montrons que GraIL surpasse les lignes de base d'induction de règles existantes dans le cadre inductif. Nous démontrons également des gains importants obtenus en assemblant GraIL avec diverses méthodes d'intégration de graphe de connaissances dans le cadre transductif, mettant en évidence le biais inductif complémentaire de notre méthode.

Contents

List of Figures

List of Tables

1

Introduction

Recent machine learning models have proved to be exceptional in recognizing patterns within high-dimensional features representing an individual data-point (object) [1, 2, 3]. In addition to tasks based on traditional labelled i.i.d. data (e.g., object classification, machine translation, and speech recognition), there are numerous important relational tasks where different objects interact with each other (e.g, scene understanding [4, 5], question answering [6, 7], and dialogue understanding). Such relational tasks require complex reasoning abilities beyond recognizing patterns in individual object features. The primary focus of this work is to study machine learning models that are specifically designed for such relational reasoning tasks.

In this work, we focus on *knowledge graphs*, which are a collection of facts about entities and relations among them. They are used in a number of applications, such as semantic parsing [8, 9], named entity disambiguation [10, 11], information extraction [12, 13], and question answering [6, 7]. While relational reasoning is implicitly vital for many tasks like scene understanding and natural language question answering, those tasks intertwine linguistic/visual (perceptual) aspects of learning with the relational reasoning abilities of the model. Knowledge graphs, being explicitly structural, let us isolate the relational reasoning from perceptual learning. This disentanglement of tasks helps us better study the innate reasoning abilities of our models and thus enable us to improve explicitly the reasoning capabilities of these models.

Predicting missing facts in knowledge graphs—usually framed as relation prediction between two entities—is a widely studied problem in statistical relational learning [14]. It requires reasoning over existing relational facts among the entities to infer new relations (Figure 1.1). The most dominant and successful paradigm, in recent times, has been to learn and operate on latent representations (i.e., embeddings) of entities and relations. These methods condense each entity's neighborhood connectivity pattern into an entity-specific low-dimensional embedding, which can then be used to predict missing relationships [15, 16, 17, 18]. Embedding-based methods have enjoyed great success by exploiting such local connectivity patterns. However, all these approaches inherently assume a fixed set of entities in the graph – an assumption that is generally referred to as the *transductive* setting (Figure 1.1b) [19].

In this thesis, we instead study the *inductive* setting, where we must predict relationships between new entities after training. This is an important problem because many real-world knowledge graphs are ever-evolving with new nodes or entities being added over time–e.g., new users and products on e-commerce platforms or new molecules in biomedical knowledge graphs. The ability to make predictions on such new entities without expensive re-training or entity resolution is essential for production-ready machine learning models. Moreover, as we discuss below, this inductive relation prediction task requires learning the logical rules underlying a knowledge graph, making it a natural testbed for investigations into relational reasoning.

In order to approach this task, we build upon the success of graph neural networks (GNNs) [20, 21, 22, 23, 24, 25, 26, 27]. GNNs are a popular and powerful approach, but we are the first to test them on this task and study their logical reasoning abilities in a multi-relational setting.

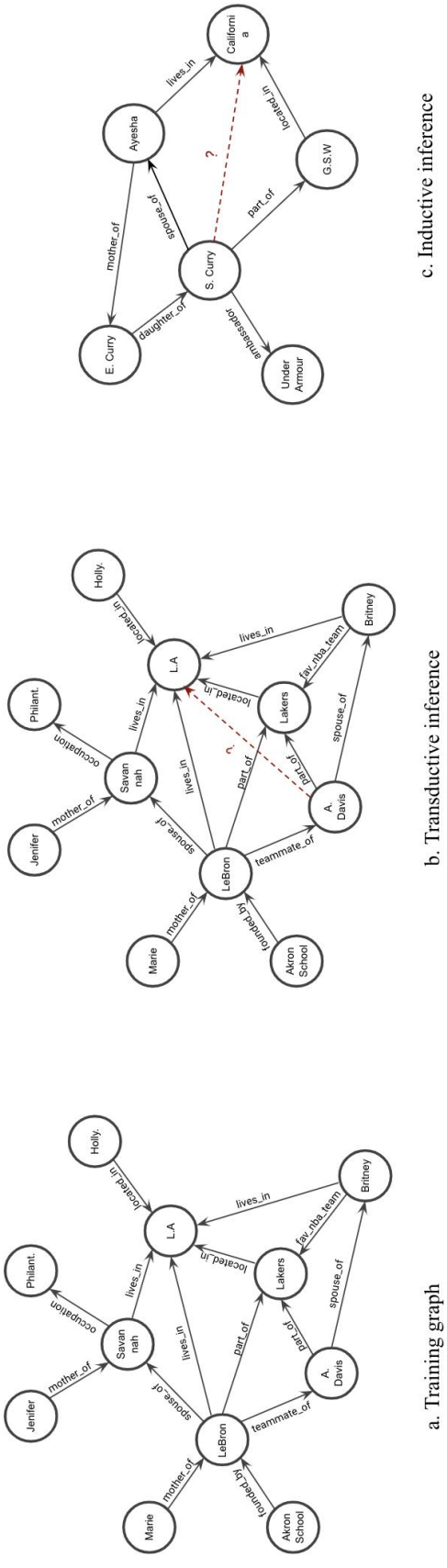


Figure 1.1: Illustration of transductive and inductive settings for relation prediction in knowledge graphs.

1.1 Problem statement

In this work, we seek to answer two questions: 1) how to learn the entity-independent relational semantics underlying the knowledge graphs, in order to enable predictions on new entities; 2) how do such relational semantics compliment the entity-specific information encoded by the embedding-based methods.

1.1.1 Learning entity-independent relational semantics

One can extract probabilistic logical rules (Horn clauses) from a Knowledge Graph. For example, from the knowledge graph shown in Figure 1.1a one can derive the simple rule,

$$\exists Y.(X, \texttt{spouse\_of}, Y) \wedge (Y, \texttt{lives\_in}, Z) \rightarrow (X, \texttt{lives\_in}, Z), \tag{1.1}$$

which says, '*A person lives in the same place as their spouse.*' We can imagine knowledge graphs to be composed of several such underlying logical rules [28, 29]. In other words, we hypothesize that most of the facts in the knowledge graph can be distilled into logical rules that capture the essence of knowledge present in them. We refer to such underlying rules as *relational semantics* since they are patterns of interactions only between the relations independent of any entities. These *relational semantics* can be used to infer missing links in the knowledge graph. For example, this rule can predict the relation (`A.Davis`, `lives_in`, `L.A`) in Figure 1.1b.

The relation prediction task can hence be interpreted as a two-step process: 1) efficiently inducing and expressing the underlying logical rules, 2) using these rules to infer the missing link(s). Note that these two steps involve two distinct problems of 1) rule induction, and 2) inference, making it all the more challenging. Moreover, such relational semantics go beyond just knowledge graphs. Explicitly identifying and learning such rules can aid in many different domains like better scene understanding

beyond raw pixels [30], navigating multi-doc textual information [31, 32], generalizing across different RL environments, etc.

With embedding-based methods encoding the local neighborhood information into the entity-specific embeddings, we can imagine the *relational semantics* being implicitly captured along with many other latent features that the model learns. For example, in Figure 1.1a, the embeddings of `LeBron` and `A.Davis` will contain the information that they are both part of the `Lakers` organization, which could later be retrieved to predict that they are teammates. Similarly, the pattern that anyone closely associated with the `Lakers` would likely live in `L.A` could be encoded in the embedding space. Although *relational semantics* by the virtue of their inherent nature are entity independent, embedding-based methods fundamentally depend on entity-specific embeddings as a medium to capture them. These methods ground themselves to the entities present in the training set and therefore can not generalize to any new nodes that show up during inference time [33, 34]. For example, an embedding-based model trained on the knowledge graph in Fig 1.1a can not make predictions on the knowledge graph in Fig 1.1c since it does not have an embedding for the new entities `S.Curry` and `California`.

In contrast, one of the key advantages of learning entity-independent relational semantics is the *inductive* ability to generalise to unseen entities [33, 34]. They can be used to make predictions on any set of entities irrespective of their presence during training. For example, the rule in Equation (1.1) can naturally generalize to the unseen knowledge graph in Fig 1.1c and predict the relation (`S.Curry`, `lives_in`, `California`). This *inductive* ability is important since many real world situations require relation prediction on dynamic or previously unseen knowledge graphs (e.g., for question answering, dialogue, or e-commerce applications) [35, 21]. Using such an inductive model, one could also transfer knowledge from one domain to another as long as the relational semantics remain the same. For example, a model trained on knowledge graph derived from one e-commerce platform could be used to make

meaningful predictions on another e-commerce platform (with entirely different users and products) without having to re-train the model.

Existing state-of-the-art naturally inductive methods are statistical rule-mining methods [29, 28]. All these methods learn probabilistic rules; i.e. each rule is associated with a probability indicating the model's confidence in that rule. Such probabilistic rules better model statistically complex and noisy data. However, it is challenging to learn rule such probabilistic rules since it involves: 1) search of a potentially vast *discrete* space of rules and 2) learning associated confidence scores in the continuous space. In this work we introduce a graph neural network [36, 37] based completely differentiable approach, GraIL (Graph Inductive Learning), to not only implicitly encode the probabilistic rules but also effectively compose multiple such rules to do better inference. In our approach, instead of learning entity-specific embeddings we learn to predict relations solely from the subgraph structure around a candidate relation.

1.1.2 Improving embedding-based methods

Despite the importance and significance of learning entity-independent relational semantics, embedding-based methods have their own strengths in the transductive setting. In particular, encoding entity-specific information into dedicated latent variables (i.e. embeddings) can capture any global patterns in the data that are otherwise difficult to capture by just structural information. Meilicke et al.[29] studies several benchmark knowledge graphs to highlight that embedding-based methods outshine rule-based methods in structurally noisy data (i.e., if the underlying rules are not very consistent). Toutanova et al.[38] and Nickel et al.[39] have given theoretical and empirical insights into the complimentary strengths of the two approaches. In particular, Nickel et al.[39] show that embedding methods can be inefficient when the relational data consists of many strongly connected components, and for such rela-

tions modelling simple local and quasi-local graphs patterns is often useful. Given our approach also works entirely on structural cues, we believe that it embodies an inductive bias that is complementary to all the embedding-based approaches and can hence help improve the existing state-of-the-art embedding-based models. This motivates us to investigate if our approach can improve the current state-of-the-art transductive models for link prediction in knowledge graphs.

Moreover, one of the key advantages of graph neural networks is their natural ability to use any available node features/attributes [37]. If there is textual or categorical features associated with the nodes of a graph (for example e-commerce knowledge graphs with product descriptions and user preferences, or user meta data in social networks, or molecular properties in biomedical knowledge graphs), the approach proposed in this thesis (GraIL) could potentially leverage that information for making better predictions. To that end, we explore GraIL's ability to leverage the information from pre-trained entity embeddings.

1.2 Thesis statement

We propose a relational learning approach that is extremely parameter efficient (and hence scalable), expressive, and naturally inductive. In particular, we present a Graph Neural Network (GNN) [40, 41] framework (GraIL: Graph Inductive Learning) that has a strong inductive bias to learn entity-independent relational semantics. It naturally generalizes to unseen nodes, as the model learns to reason over subgraph structures independent of any particular node identities. We provide theoretical analysis of representational capacity of GraIL and prove that it can represent logical rules of the kind presented above (e.g., Equation (1.1)) and effectively compose multiple such rules.

In order to test models with inductive capabilities we introduce a series of benchmark tasks for the inductive relation prediction problem. Existing benchmark datasets

for link prediction in knowledge graphs are set up for transductive reasoning, i.e., they ensure that all entities in the test set are present in the training data. Thus, we construct several new inductive benchmark datasets by carefully sampling subgraphs from diverse knowledge graph datasets. Extensive empirical comparisons on these novel benchmarks demonstrate that GraIL is able to substantially outperform state-of-the-art inductive baselines. Further, we demonstrate the complimentary structural inductive bias of GraIL by exploring several ensembling strategies with existing state-of-the-art transductive methods.

1.3 Statement of contribution

The main contributions of the thesis are as follows:

- Present an alternate paradigm of entity-independent relational semantics for knowledge graph completion and introduce the inductive problem.
- Provide a comprehensive survey of existing methods capable of performing inductive relation prediction on knowledge graph completion.
- Introduce novel benchmark datasets for inductive knowledge graph completion.
- Develop a novel algorithm based on GNNs, GraIL, that learns entity-independent relational semantics of a knowledge graph and is naturally inductive.
- Provide theoretical analysis of expressive power of this new algorithm and its' connections with first order predicate logic.
- Detailed experimental comparison with previous inductive models showing
  - The superior performance of GraIL in the inductive setting.
  - The parameter efficiency of GraIL with respect to existing differentiable inductive methods.

- Ensembling strategies showcasing the complementary inductive bias of GraIL with respect to existing knowledge graph embedding methods.
- Highlighting the natural ability of GraIL to incorporate pre-trained node features, when available.

This thesis is based on a paper with the same title— 'Inductive Relation Prediction by Subgraph Reasoning'—that is to appear in the proceedings of *Thirty-seventh International Conference on Machine Learning, 2020*. The author of this thesis (Komal Teru) is also the lead author of this publication. We are thankful to the co-author, Etienne Denis, who assisted in the experimental design and setup. The key author contributions of the individual chapters of this thesis—and how they relate to this publication—are clarified in the next section.

1.4 Outline of the thesis

The remaining thesis is organized as follows:

In Chapter 2, we provide the necessary background on knowledge graphs introducing the main task of focus, *relation prediction*. We describe the modelling framework, training mechanism and evaluation protocols of different approaches to this task with main focus on embedding-based models. This set of models constitute the baselines in some of our experimental studies. The contents of this chapter form the foundations of the task setup studied in this work.

In Chapter 3, we present the unified message passing framework of graph neural networks describing various strategies developed in recent literature. We further introduce GNN-based models for relation prediction in multi-relational graphs. The contents of this chapter form the foundations for our proposed model.

In Chapter 4, we present the core contributions of this work: a novel GNN-based relation prediction model. We provide a theoretical analysis of the proposed model,

GraIL, and establish it's connections to first order predicate logic. We also discuss the computational complexity and scalability of the proposed model. The model and the theorems proposed in this chapter were developed with assistance and guidance of the supervisor while the author of this thesis is solely responsible for formulating the proofs of the said theorems.

In Chapter 5, we present extensive experimental results to showcase the modelling capacity of the proposed framework. In particular, we present experiments in the inductive setting, transductive setting and several additional analysis to substantiate our theoretical findings in Chapter 4. The collaborator, Etienne Denis, assisted in the efficient pre-processing of the data used in the experiments. All the analysis, experiments and ablation studies reported in this chapter were conducted exclusively by the author of this thesis.

In Chapter 6, we conclude our findings with a brief discussion of the limitations of our method. We also point out to interesting new directions of research that this work opens up in the context of current frontiers of logical reasoning and graph neural networks.

2

Knowledge Graphs

In this chapter we introduce knowledge graphs (KGs), the main object of study in this thesis. We describe the relation prediction task along with many other tasks that can be performed on knowledge graphs. We explain the modelling framework, training mechanism and evaluation protocols of different approaches to this task with a main focus on embedding-based models. Towards the end, we briefly discuss extensions beyond the standard static knowledge graphs and how this these extensions connect to our current method.

Knowledge graphs model complex relational information between different entities. They represent facts as binary triplets of the form *(subject, predicate, object)* where *subject* and *object* are the entities and the *predicate* is the relation between them. A large collection of these triplets capture complex relational information about the world. For example, the information

> *LeBron, along with his teammate A. Davis, play basketball for L.A.Lakers team located in Los Angeles.*

can be represented as the following set of triplets

subject	*predicate*	*object*
(*LeBron*	*teammate_of*	*A. Davis*)
(*LeBron*	*part_of*	*Lakers*)
(*A. Davis*	*part_of*	*Lakers*)
(*Lakers*	*located_in*	*Los Angeles*)

All the triplets (facts) together can be represented as a multi-relational graph. For example, the above set of facts, along with many more, are curated in the illustrative knowledge graph in Figure 1.1a.

Completeness and accuracy are important aspects of knowledge graphs, which determine their usefulness in downstream applications. These aspects are heavily influenced by the construction methodology of knowledge graphs. This can be mainly categorized into three different methodologies:

1. *Curated approach*: The facts are curated by a closed group of domain experts. The WordNet dataset [42] used in this thesis is an example of curated knowledge graph. This way of knowledge graph construction results in highly accurate and reliable facts. However, it requires significant human intervention and is not scalable to build large-scale knowledge graphs.

2. *Collaborative and automated semi-structured approach*: The facts are collected by an open group of volunteers who follow a common schema. This mitigates the scalability issue of curated approach but may result in slightly inaccurate and incomplete knowledge graphs. To further enrich the knowledge graph with more information, facts could be automatically extracted from structured text on the web (e.g., infoboxes on Wikipedia) using pre-defined rules. The FreeBase dataset [15] used in this thesis was constructed using both these strategies.

3. *Automated unstructured approach*: In this approach, the facts are extracted from unstructured text using machine learning and natural language processing

techniques. This enables construction of very large-scale knowledge graphs by leveraging all the free-flowing text available on the web. However, the quality of the resulting facts is limited by the abilities of the underlying text processing algorithms. For example, there can be many duplicate entities due to the lack of accurate entity resolution abilities in the text extractor. Many facts collected can be erroneous or contradictory because of the inconsistent nature of the unfiltered information on the web. The NELL [43] dataset used in this thesis is a continually growing knowledge graph constructed by regularly crawling the web and adding new information.

Tasks on knowledge graphs

There are a number of tasks that can be typically performed on knowledge graphs.

Relation prediction Sometimes referred to as *link prediction* or *entity prediction*, it refers to predicting the existence (or the probability of correctness) of a relation between two entities. This task manifests itself is various forms. For example, it could be formulated as predicting the tail entity in a triplet, $(h, r, ?)$. A common practice to evaluate this is to score all entities as a potential prediction and record the rank of the true entity. The prediction tasks $(h, ?, t)$ or $(?, r, t)$ can be carried out in a similar manner. Relation prediction can also be formulated as *triplet classification* where given a triplet (h, r, t) the task is to predict the correctness of that fact. While the actual formulation of *relation prediction* depends on the application at hand, all these different formulations share a common learning problem. They all require the model to reason over the facts present in the knowledge graphto find evidence for missing facts. This task is important for various applications like question answering [6, 7], recommendation systems [44], knowledge graph completion [15, 18], etc.

Entity and relation resolution This refers to identifying which objects in the knowledge graph refer to the same underlying entity or relations. This task is of importance especially in knowledge graphs constructed automatically from unstructured text. For e.g., a text processor might add two different triplets—("King James", "born in", "Akron") and ("LeBron James", "place of birth", "Akron")—without recognizing that the first triple refers to the same person as the second triple, nor knowing that "born in" means the same thing as "place of birth". Resolving such duplicacies would avoid unnecessary increases in the size of knowledge graph and facilitate meaningful additions [45].

Entity classification This refers to predicting properties of entities in the knowledge graph based on their relational information. For e.g., the task could be to classify entities into different semantic categories such as person, place, occupation, etc. In many cases, such tasks are seeded by a small set of labelled entities which help us learn to predict the the properties of all other nodes in the graph [20]. In absence of such a labelled seed set, unsupervised clustering techniques help the model discover underlying properties of different nodes.

The focus of this thesis is the *relation prediction* task. There are largely two lines of work that address this task – embedding-based methods and rule-based methods.

2.1 Embedding methods

Let $\mathcal{G}(\mathcal{T}; \mathcal{E}, \mathcal{R})$ be a knowledge graph where $\mathcal{E}$ and $\mathcal{R}$ denote the set of entities and relations, respectively; $\mathcal{T}$ denotes the set of triplets (facts) (u, r, v) with $u, v \in \mathcal{E}$ and $r \in \mathcal{R}$. Although *relation prediction* takes many forms (as described earlier), the fundamental problem across all the variations is to score a given triplet, (u, r, v). This score corresponds to model's belief of the correctness of this fact. The scoring mechanism of all embedding-based methods can be described by an *encoder-decoder*

framework [37]. This framework is built around two mapping functions: an *encoder*, which maps the entities ($\mathcal{E}$) and the relations ($\mathcal{R}$) of $\mathcal{G}$ to a low-dimensional vector (embeddings), and a *decoder*, which scores a given triplet using the respective entity and relation embeddings. In particular, the *encoder* can be sub-categorized into *entity encoder* and *relation encoder* functions,

$$\texttt{ENT\_ENC} : \mathcal{E} \rightarrow \mathbb{R}^d$$

$$\texttt{REL\_ENC} : \mathcal{R} \rightarrow \mathbb{R}^d$$

which map entities and relations of $\mathcal{G}$ to a d-dimensional vector, respectively [1]. The *decoder* is a function that maps the entity and relation embeddings of a given triplet and to a real number,

$$\texttt{DEC} : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}.$$

This real number corresponds to the score indicating the probability of the given triplet. *The goal is to optimize the encoder and decoder parameters to score the triplets that are known to be true $\mathcal{G}$ higher than the ones that are assumed to be false.* The diversity of methods in the literature stem from the use of different encoders and decoders. Here, we describe a representative set of decoder functions and methods that employ these with a simple embedding look-up encoders i.e., `ENT_ENC` and `REL_ENC` simply return an embedding from a fixed dictionary of entity and relation embeddings.

2.1.1 Decoder functions

A vast variety of methods are characterized by the *decoder function* they use. In fact, the *decoder functions* embody the inherent inductive biases of the model.

[1] Note that it is very possible for these encoders to embed entities and relations to different vector spaces. Following most widely used methods, we simplify them to map to the same vector space. Also, as we will see some methods map them to complex space rather than the real space.

Distance-based decoders

A wide class of methods exploit distance-based scoring functions. They compute the likelihood score of a fact as the distance between the head and the tail entity embedding, usually *after* a geometric transformation is carried out by the relation embedding.

TransE and extensions A wide range of decoders represent relations as translations in the embedding space. Bordes et al.'s TransE [15] model initiated such an approach by defining the decoder as

$$\texttt{DEC}(u, \tau, v) = -||\texttt{ENT\_ENC}(u) + \texttt{REL\_ENC}(r) - \text{ENT\_ENC}(v)|| \tag{2.1}$$

$$= -||\mathbf{z}_u + \mathbf{r}_\tau - \mathbf{z}_v||, \tag{2.2}$$

where $\mathbf{z}_u, \mathbf{z}_v \in \mathbb{R}^d$ correspond to the embeddings of the entities u and v, and $\mathbf{r}_\tau \in \mathbb{R}^d$ corresponds to the embedding of relation τ. The interpretation of this decoder is that the head entity and the tail entity are connected by a certain relation if translating the head embedding by the relation vector brings it closer to the tail embedding. This intuition was based on similar findings in the context of word embeddings [46]. This model is relatively simple but proved to be a strong baseline in many applications. Many extensions to TransE, generally referred to as TransX models [47, 48, 49, 50], have been proposed. All such decoders can be generalized as

$$\texttt{DEC}(u, \tau, v) = -||g_{1,\tau}(\mathbf{z}_u) + \mathbf{r}_\tau - g_{1,\tau}(\mathbf{z}_v)||, \tag{2.3}$$

which adds the added flexibility to learn relation-specific transformations, $g_{i,\tau}$, to the entity embeddings. For example, one such extension—TransH [47]—projects the head and tail embeddings to relation-specific hyperplanes before translating and computing the score. While many such translational methods have been proposed, they have fundamental limitations in their expressive power. For example, they cannot represent symmetric relations by a non-zero translating vector representation.

RotatE Building on the same idea of measuring a triplet plausibility as the distance between head and tail embeddings after a certain geometric operation, Sun et al. propose RotatE model [18] which *rotates* the head embedding instead of translating. In particular, the decoder function is given by

$$\texttt{DEC}(u, \tau, v) = -||\mathbf{z}_u \circ \mathbf{r}_\tau - \mathbf{z}_v||, \tag{2.4}$$

where $\mathbf{z}_u, \mathbf{z}_u, \mathbf{r}_\tau \in \mathcal{C}^d$ all are complex-valued embeddings, and $\circ$ denotes the Hadamard product. Each dimension of relation embeddings in the complex plane, $\mathbf{r}_\tau$, are constrained to have unit norm, i.e., $|\mathbf{r}_\tau[i]| = 1$. This constraint enforces the Hadamard product of head and relation embeddings to represent rotation of each dimension of head embedding by the respective dimension of the relation embedding. The rotation operation allows RotatE to model a wide range of relation types including symmetric relations, making it one of the strongest methods for relation prediction in knowledge graphs.

Similarity-based decoders

Many methods, instead of using distance-based measures, exploit similarity-based scoring functions. They measure the plausibility of a triplet by matching the latent semantics of the constituent entities and relation.

RESCAL and DistMult RESCAL [51] is one of the older methods which uses similarity-based scoring functions. It captures the latent semantics of the entities in a d-dimensional vector space, and relations are represented with matrices that model pairwise interactions between the latent factors of head and tail entities. The scoring function is given by

$$\texttt{DEC}(u, \tau, v) = \mathbf{z}_u^T \mathbf{M}_\tau \mathbf{z}_v = \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} \mathbf{z}_u[i] \times \mathbf{M}_\tau[i, j] \times \mathbf{z}_v[j], \tag{2.5}$$

where $\mathbf{z}_u, \mathbf{z}_v \in \mathbb{R}^d$ denote the latent representations of the entities u and v, and $\mathbf{M}_\tau \in \mathbb{R}^{d \times d}$ denotes the matrix associated with relation τ. DistMult [52] simplifies

RESCAL by restricting the relation representation to be a diagonal matrix. In other words, a relation τ is represented by a vector $\mathbf{r}_\tau$ such that the matrix $\mathbf{M}_\tau = \text{diag}(\mathbf{r}_\tau)$. The resulting scoring function is simplified to

$$\texttt{DEC}(u, \tau, v) = \mathbf{z}_u^T \text{diag}(\mathbf{r}_\tau)\mathbf{z}_v = \sum_{i=0}^{d-1} \mathbf{z}_u[i] \times \mathbf{r}_\tau[i] \times \mathbf{z}_v[i]. \tag{2.6}$$

This simplification allows for scaling to larger knowledge graphs since now the number of parameters linearly scale with number of relations. However, the expressive power is now limited in that DistMult can capture pairwise interactions between latent factors of head and tail entities only across the same dimension. Moreover, with this simplification the model loses the sense of head and tail, i.e.,

$$\texttt{DEC}(u, \tau, v) = \texttt{DEC}(u, \tau, u) \tag{2.7}$$

This deprives the model of the ability to distinguish asymmetric relations.

ComplEx To better model asymmetric relations, Trouillon et al. proposed ComplEx [16] which extends DistMult to represent the entities and relations in the complex space instead of real space. The scoring function is given by

$$DEC(u, \tau, v) = \text{Re}(\mathbf{z}_u^T \text{diag}(\mathbf{r}_\tau)\overline{\mathbf{z}_v} = \text{Re}(\sum_{i=0}^{d-1} \mathbf{z}_u[i] \times \mathbf{r}_\tau[i] \times \overline{\mathbf{z}_v}[i]), \tag{2.8}$$

where $\mathbf{z}_u, \mathbf{z}_v, \mathbf{r}_\tau \in \mathbb{C}^d$ are complex valued embeddings and Re(.) denotes the real component of a complex vector. Since we take the complex conjugate, $\overline{\mathbf{z}_v}$, of the tail entity embedding, this scoring function is asymmetric and can accommodate asymmetric relations.

Many other methods and approaches have been proposed for scoring triplets in knowledge graphs [53, 54, 55, 56]. While all these methods employ a straightforward embedding look-up encoders, one can imagine pairing the described decoder functions with more sophisticated encoders. For instance, the entity encoder can be a function of the neighboring entities and relations [57]. Similarly, the relation encoder can be a function of the all the entities they connect to in the knowledge graph[58].

As we will see in the next chapter, Graph Neural Networks are a natural choice to encode neighborhood information and many approaches have recently been proposed to leverage their representational capacity [59, 60, 61] for the knowledge graph relation prediction task.

2.1.2 Training regime

The goal of training these models is to optimize the encoder and decoder parameters to score the triplets that are known to be true in $\mathcal{G}$ higher than the ones that are assumed to be false. One of the straightforward ways to do this, as done in the case of simple graphs, is to train our model parameters with a reconstruction loss as follows

$$\mathcal{L} = \sum_{u \in \mathcal{E}} \sum_{\tau \in \mathcal{R}} \sum_{v \in \mathcal{E}} ||\texttt{DEC}(\texttt{ENC}(u), \texttt{ENC}(\tau), \texttt{ENC}(v)) - \mathcal{A}[u, \tau, v]||^2, \tag{2.9}$$

where $\mathcal{A} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{R}| \times |\mathcal{E}|}$ denotes the adjacency tensor of the knowledge graph. In the simplest case, this is a binary tensor with $\mathcal{A}[u, \tau, v] = 1$ if and only if $(u, \tau, v) \in \mathcal{E}$. This loss is usually optimized using stochastic gradient descent [62].

However, there are two key disadvantages of training our models this way.

1. The loss in Equation (2.9) is very expensive to compute. It requires $\mathcal{O}(|\mathcal{E}|^2|\mathcal{R}|)$ operations, which is prohibitive in many graphs. Moreover, since many knowledge graphs are very sparse, i.e., $|\mathcal{E}| << |\mathcal{E}|^2|\mathcal{R}|$, we would ideally want a loss function that only takes $\mathcal{O}(|\mathcal{E}|)$.

2. In a binary adjacency tensor, $\mathcal{A}$, by definition a value of 1 indicates the corresponding edge being true and a 0 indicates the edge being false. When trying to reconstruct $\mathcal{A}$, we are inherently assuming that all the edges that are not present in the knowledge graph are false. This is commonly referred to as the *closed world assumption* (CWA) [14]. However, in relation prediction the goal it to predict edges that are missing from the current knowledge graph. Thus, while CWA may be useful in certain tasks (e.g. entity clustering) it goes against the

fundamental premise of relation prediction. Open world assumption (OWA) [14] that missing triplets are neither false nor true is more apt to relation prediction.

Cross entropy loss with negative sampling

One of the strategies to overcome these challenges, inspired by the way word embeddings are trained [63], is to treat our task as binary classification task and using cross entropy loss with negative sampling as follows

$$\mathcal{L} = \sum_{(u,\tau,v)\in\mathcal{E}} -\log(\sigma(\text{DEC}(u,\tau,v))) - \gamma \sum_{v_n\in P_n(\mathcal{E};u)} [\log(\sigma(-\text{DEC}(u,\tau,v_n))], \tag{2.10}$$

where σ denotes the logistic function, $P_n(\mathcal{E};u)$ denotes a 'negative sampling' distribution over the set of entities $\mathcal{E}$ (which might depend on u), and γ is a hyperparamter [20, 21]. As in the standard binary cross-entropy loss, the first term is log-likelihood of the true triplets present in the knowledge graph and the second term corresponds to the expected log-likelihood of 'false' triplets that are randomly sampled from the KG. Random sampling of negative triplets is a reasonable strategy in the context of KGs because of their inherent sparse nature, a point described in detail in later sections. Thus, the models are in effect trained to score the triplets present in the KG higher than the ones not present.

Max-margin loss with negative sampling

An alternate loss function is what is commonly referred to as the *max-margin loss* [15, 18]:

$$\mathcal{L} = \sum_{(u,\tau,v)\in\mathcal{E}} \sum_{v_n\in P_n(\mathcal{E};u)} \max(0, -\text{DEC}(u,\tau,v) + \text{DEC}(u,\tau,v_n) + \Delta). \tag{2.11}$$

Unlike the cross-entropy loss whose objective is to learn to predict directly a label, the objective of max-margin loss is to predict relative distances between inputs. In particular, optimizing to minimize the loss in Equation (2.11) tries to score the positive triplet (u,τ,v) higher by a margin of Δ (a hyperparameter) than the sampled negative

triplet (u, τ, v_n). This is further closer to the open-world assumption of knowledge graphs in that we don't necessarily classify the negative triplets as 'false'.

Negative sampling

Cross-entropy and max-margin loss functions both need negative triplets to score against positive triplets. In most cases when negative triplets are not explicitly known they are obtained by some sampling scheme. The way in which the negative triplets are sampled can have a large impact on the learned model [64]. The most common strategy to sample negative triplets is to replace the tail entity of a true triplet with a uniformly sampled random entity [15]. The false negatives that are sometimes obtained in this procedure, i.e. when $(u, \tau, v_n) \in \mathcal{E}$, are removed [16]. Given the inherent sparse nature of knowledge graphs the probability of obtaining such false negatives is very low and hence this sampling mechanism is both feasible and practical.

Such naive sampling scheme might result in many 'easy' negative triplets that are not very informative of the underlying semantics. For example, although the triplet `(Ottawa, capitol_of, J. Trudaeu)` is false and useful to learn the type compatibility of relations and entities, it doesn't hold as much semantic value as the type compatible negative triplet `(Ottawa, capitol_of, India)`. The latter is in some sense a more 'difficult' negative triplet and more revealing. If the ontology of a knowledge graph is known, type consistent negative triplets could be sampled for learning better models. Sun et al.[18] further propose an approach to select challenging negative samples using an adversarial network.

It is worth noting that without loss of generality we have thus far replaced only the tail entity to generate negative samples. In practice, we consider negative samples generated by replacing either the head or the tail entity, i.e., (u, τ, v_n) or (u_n, τ, v). This is important since knowledge graphs are directional and replacing only the tail can lead to undesired biases.

2.1.3 Evaluation regime

There are two widely followed evaluation protocols for *relation prediction*: 1) classification metric [65] and 2) ranking metric [15, 18]. The classification metric mainly reflects a models' performance on the triplet classification task and the ranking metric reflects performance on the entity prediction task.

Classification metric The triplet classification task is to simply score the plausibility of a given triplet (u, r, v). This is useful in settings where the goal is to verify the validity of candidate facts, rather than proposing new facts. For example, a biologist may want to test the plausibility of a hypothesis using a biomedical knowledge graph. To evaluate the ability of a model to do this task, we sample random negative triplets (in a similar way as we did during training—by replacing the head or the tail of a positive triplet) and evaluate the area under the curve (AUC) for the model. With unequal number of positive and negative triplets, area under the precision-recall curve (AUC-PR) is more informative.

Ranking metric An alternate formulation of the relation prediction task is to predict a missing entity from a triple, i.e., predicting a missing head $(?, r, v)$ or a missing tail $(u, r, ?)$. This is useful in settings where the goal is to propose new facts, e.g., in an e-commerce knowledge graph, we may want to infer missing attributes of customers. This is evaluated by scoring and ranking all potential entities that could complete the triple. The set of potential entities is typically referred to as the *candidate set*. We want the model to rank the true head or tail higher than other potential answers. A common set of metrics reported to reflect this ability are Hits@k and Mean Reciprocal Rank (MRR). Hits@k denotes the fraction of test triplets for which the true entity was ranked in the top k predictions. MRR, as the name suggests, is the mean of the reciprocal of rank of the true entity across all test triplets. While Hits@k reflects the fraction of the test triplets that the model gets right, MRR gives

a more holistic sense of the models' performance on all the test triplets.

Note that both these metrics require negative sampling and the evaluation is very sensitive to the negative triplets sampled [64]. For classification metrics, all the strategies from the training regime can be directly applied. In particular, strategies to sample 'harder' negatives can be employed for more stringent evaluation of the models' ability to predict the likelihood of triplets. For ranking metrics, the choice of the *candidate set* directly effects the model evaluation. In the simplest case, all the entities in the knowledge graph make the candidate set. But very often, much smaller and more relevant relation-specific candidate sets can be derived using type constraints and meta-data available [66]. For example, tail of the relation `mother_of` can only be of the type `person`. Hence, ranking the true son of the head entity among a candidate set consisting entities of only type `person` is more meaningful than ranking the true son among, say, entities of type `place`.

2.2 Rule-based methods

While the embedding methods try to capture latent features of the knowledge graphs, rule-based methods capture observed rules from the graph. Many statistical rule-mining approaches induce probabilistic logical rules by enumerating statistical regularities and patterns present in the knowledge graph [67, 68, 28, 29]. These induced rules are then used to predict missing relations. These methods are naturally interpretable but face scalability challenges due to the large search space of discrete rules. Recently, many methods have tried to combine embedding based methods with logical rules. Q. Wang et al.[69] utilizes logical rules to refine the predictions made using embeddings. Wei et al.[70] propose a similar idea that combines rules and embedding models via Markov Logic Networks [71]. Many works [72, 73, 74, 75, 76, 77] have also shown effective approaches to jointly optimize the embeddings and logical rules. They all learn entity and relation embeddings subjecting them to some constraints

derived from first-order logical rules (Horn clause). This incorporates the general first-order logical rules in entity and relation representations, which enhance the predictive abilities of the embeddings.

Recent works have proposed approaches to leverage the inference capabilities of Markov Logic Networks and efficiency of embedding methods by jointly optimizing them using variational expectation-maximization algorithm [78, 79]. While these methods have their own merits, all of them require already extracted rules from the knowledge graph which are sometimes quite expensive to extract. On this front, the approach presented in this thesis, GraIL, elegantly blends the rule extraction and inference steps bypassing the need to explicitly extract rules.

2.3 Beyond Static knowledge graphs

The methods we have seen so far are all designed for static knowledge graphs, i.e., graphs with fixed set of entities and relations. However, real-world knowledge graphs are always evolving. There are many dimensions to the evolution of knowledge graphs– 1) new entities can be added (for e.g., in an e-commerce knowledge graph new users and items being regularly added), 2) new relation types can be added (for e.g., new biological processes being added to protein-protein interaction graphs) 3) the interactions between the entities may change with time. We will refer to these evolving KGs as Dynamic knowledge graphs. Depending on the domain the dynamics can be a complex combination of all the above 3 factors or just one of them.

New entities Many methods have been proposed to handle new entities in knowledge graphs. The new entities are sometimes referred to as out-of-vocabulary entities. Several works consider attributed knowledge graphs and induce the embeddings for new entities from their text/image descriptions [80, 81, 82]. For unattributed KGs P. Wang et al.[83] and Hamaguchi et al.[84] learn to generate embeddings for new

entities by aggregating information from embeddings of their neighboring entities. This requires the new nodes to be surrounded by known nodes. While these methods design explicit strategies to accommodate new entities, the approach proposed in this thesis to learn entity-independent relational semantics makes our method, GraIL, unaffected by such out-of-vocabulary entities. In fact, GraIL can make predictions on entirely new graphs with new entities as long as the underlying relational semantics (logical rules) remain the same.

New relations It is also possible to have new relations being added to the knowledge graph. Even more prevalent is the notion of long tail in relation frequency distribution [66]. To be precise, it is common to have many relations which only have a few triplets in the knowledge graph. Many methods have been proposed to tackle this issue [66, 85, 86]. All of these approaches frame the problem as a few-shot link prediction and adapt a meta-learning framework [87]. This meta-learning approach gives the models the ability to generalize to new relations.

Temporal knowledge graphs Another aspect of dynamic knowledge graphs is the temporal nature of the facts. For example, (`Obama`, `president_of`, `U.S.A`) is only valid for a certain period of time and the fact changes after that period. Not only that many applications require the models to make time-dependent predictions [88, 89, 90], the ability to exploit and account for such temporal dynamics of a knowledge graph can lead to improved performance of traditional relation prediction. Many methods have recently been proposed to address this challenge [91, 92, 93].

3

Graph Neural Networks

In the previous chapter, we have seen an overview of knowledge graphs and various methods used for the relation prediction task. In this chapter, we introduce a new class of methods—Graph Neural Networks (GNNs)—which operate on any graph structured data in general and have proved to be a promising approach for various tasks on knowledge graphs. We first introduce the general formalism of GNNs for simple graphs before giving an overview of current GNN-based approaches for knowledge graphs. We further give a brief overview of recent successes of GNNs in reasoning tasks [22, 23, 26, 27].

All the methods we have seen so far for knowledge graphs fall into the generalized encoder-decoder framework described in the previous chapter. As pointed out earlier, the encoder in those approaches is a shallow embedding look-up, i.e., it trains a separate embedding for all the entities (and relations for multi-relational graphs). GNNs provide a general formalism to design more rich and expressive encoders that leverage the graph structure more explicitly. The GNN-formalism also addresses the following key limitations of shallow encoders [37]:

- Since each entity has its own embedding in the shallow encoder approach, the number of parameters of the model scale with the number of entities in the graph. This quickly becomes prohibitive for training shallow encoders on real-world large graphs. GNNs provide an elegant way to efficiently share parameters

which also helps the regularize the models.

- The shallow encoders do not have a natural way to leverage any node features/attributes available. Such node features/attributes can be very informative with respect to the position and role of the node in a graph. GNNs can naturally leverage any such node attributes available [35, 21].

- As described before, shallow encoders are inherently transductive, i.e., they can only learn embeddings for nodes that are present during training. This approach fails to generalize to constantly evolving graphs. GNN's ability to leverage node attributes has motivated several inductive approaches on simple graphs which successfully transfer their learning to new nodes and graphs [21]. This thesis follows similar motivation but extends these works to deal with multi-relational graphs without any entity attributes.

3.1 Message passing formalism

The key idea of encoding nodes of a graph is to iteratively aggregate information from the node's neighborhood. This is commonly referred to as *neural message passing* [36]. Formally, given a graph, $\mathcal{G}(\mathcal{E}, \mathcal{T})$, induced by set of nodes $\mathcal{E}$ and consisting of the set of edges $\mathcal{T}$, along with a set of node features $\mathbf{X} \in \mathcal{R}^{|\mathcal{E}| \times d}$, we want to use this information to generate node embeddings $\mathbf{z}_u \; \forall u \in \mathcal{E}$. The node embeddings are obtained after multiple rounds of 'message passing' from the neighboring nodes. At any given iteration, l, the new node representation $\mathbf{h}_u^l$ is obtained by updating the current representation of node u, $\mathbf{h}_u^{l-1}$, with information obtained by aggregating the representation of the neighboring nodes, $\mathbf{h}_v^{l-1}$, $\forall v \in \mathcal{N}(u)$. This 'message passing'

among nodes is formally represented by

$$\mathbf{m}_{\mathcal{N}(u)}^{l} = \text{AGGREGATE}^{l}\left(\left\{\mathbf{h}_{v}^{l-1} : v \in \mathcal{N}(u)\right\}\right), \tag{3.1}$$

$$\mathbf{h}_{u}^{l} = \text{UPDATE}^{l}\left(\mathbf{h}_{u}^{l-1}, \mathbf{m}_{\mathcal{N}(u)}^{l}\right), \tag{3.2}$$

where UPDATE and AGGREGATE are arbitrary differentiable functions (i.e., neural networks) and $\mathbf{m}_{\mathcal{N}(u)}^{l}$ is the "message" that is aggregated from u's graph neighborhood $\mathcal{N}(u) : \{v|(u,v) \in \mathcal{T}\}$ [94]. Superscripts are used to distinguish the embeddings and functions at different iterations of message passing. This l^{th} round of message passing is also referred to as the l^{th} layer of the GNN.

At each iteration of message passing the AGGREGATE function takes as input the set of representations of nodes in node u's graph neighborhood $\mathcal{N}(u)$ and aggregates the information to compute a 'message' $\mathbf{m}_{\mathcal{N}(u)}^{l}$. The UPDATE function then combines the message $\mathbf{m}_{\mathcal{N}(u)}^{l}$ with the previous embedding $\mathbf{h}_{u}^{l-1}$ of node u to generate the updated embedding $\mathbf{h}_{u}^{l}$. The initial node representations at $l = 0$ are set to the input features for all the nodes, i.e., $\mathbf{h}_{u}^{0} = \mathbf{X}_{u}, \forall u \in \mathcal{E}$. If node features are not available they could be initialized to some graph-dependent structural features (e.g., node degree and/or node centrality). Another common strategy involves assigning a random unique label to every node [20, 57]. After L rounds of message passing the final node representations give the node embeddings of all the nodes in the graph, i.e.,

$$\mathbf{z}_{u} = \mathbf{h}_{u}^{L}, \forall u \in \mathcal{E}.$$

The intuition of this iterative message passing scheme is that at each iteration, every node aggregates information from its local neighborhood, and as these iterations progress each node embedding contains more and more information from further reaches of the graph. To be precise, after the first iteration each node would have aggregated information for its 1-hop neighborhood, i.e., its immediate neighbors in the graph. In the second iteration information is again aggregated from 1-hop neighborhood, however the 1-hop neighbor node representations have information aggregated

from their respective 1-hop neighborhoods. Hence, after the second iteration each node will have effectively aggregated information from their respective 2-hop neighborhoods. In summary, after L rounds of message passing, each node embedding would have the information of its L-hop neighborhood.

Given the basic intuition and the overview of the message passing formalism, the expressive capacity of a GNN now depends on the choice of AGGREGATE and UPDATE functions. These can be any arbitrary differentiable functions and many variants have been proposed and validated in the recent literature. One of the most basic versions of these functions which results in the simplest GNN is given by

$$\mathbf{m}^l_{\mathcal{N}(u)} = \sum_{v \in n\mathcal{N}(u)} \mathbf{h}^{l-1}_v \tag{3.3}$$

$$\mathbf{h}^l_u = \sigma\left(\mathbf{W}^l_{self}\mathbf{h}^{l-1}_u + \mathbf{W}^l_{neigh}\mathbf{m}_{\mathcal{N}(u)^l}\right), \tag{3.4}$$

where $\sigma(.)$ denotes element-wise non-linearity (e.g., a tanh or ReLU), and $\mathbf{W}^l_{self}, \mathbf{W}^l_{neigh}$ denote the trainable layer-specific linear transformation matrices. Here, the neighborhood message is computed by a simple addition of all the neighboring node representations. The simple UPDATE function described in Equation (3.4) gives the ability to selectively chose information from the current node representation, $\mathbf{h}^{l-1}_u$, and the neighborhood message, $\mathbf{m}^l_{\mathcal{N}(u)}$. The AGGREGATE and UPDATE functions can be generalized beyond these simple realizations to result in more powerful and expressive GNNs.

3.1.1 Generalized AGGREGATE function

Here, for the sake of completeness we give a brief overview of different generalizations of the AGGREGATE function.

Neighborhood Normalization

In the simple AGGREGATE function illustrated in Equation (3.3), we take the sum of the neighborhood node representations. This could result in disproportionate mag-

nitudes of the message vector when there exists a wide range of node degrees in the graph leading to instabilities in optimization. We could instead normalize the sum of neighborhood node representations to offset such instabilities. In fact the popular and very successful baseline GNN proposed by [20] uses the following normalized aggregate function:

$$\mathbf{m}^l_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \frac{\mathbf{h}^{l-1}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}.$$

While this normalization is motivated by spectral graph theory, a simpler version of just taking a mean of the neighborhood node representations also proved successful [21].

Although normalization in the AGGREGATE function is sometimes needed to stabilize optimization and obtain good results, it can also lead to a loss of information. For example, after normalization, it can be hard (or even impossible) to use the learned embeddings to distinguish between nodes of different degrees, and various other structural graph features can be obscured by normalization. In fact, a basic GNN using the mean aggregate function [21] operator is provably less powerful than the basic sum aggregate function in Equation (3.3) [95]. Usually, normalization is most helpful in tasks where node feature information is far more useful than structural information, or where there is a very wide range of node degrees that can lead to instabilities during optimization. As we will see, in the model proposed in this thesis the messages are passed around relatively low-degree nodes and the model is dominantly reliant on structural properties of the graph, thus motivating a sum-style aggregate function.

Neighborhood Attention

A more sophisticated way of aggregating messages from the neighborhood is to only accumulate information from the nodes that are important to the downstream task. Attention mechanisms [2] over the neighborhood nodes can facilitate this type of ag-

gregation. Inspired by this idea Graph Attention Network (GAT) [96], uses attention weights to define a weighted sum of the neighbors:

$$\mathbf{m}^l_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{h}_v, \tag{3.5}$$

where the attention weights between a pair of nodes is given by

$$\alpha_{u,v} = \frac{exp(\mathbf{a}^T[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v])}{\sum_{v' \in \mathcal{N}(u)} exp(\mathbf{a}^T[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_{v'}])}, \tag{3.6}$$

where $\mathbf{a}$ is a trainable attention vector, $\mathbf{W}$ is a trainable matrix, and $\oplus$ denotes the concatenation operation.

Attention is particularly useful for increasing the representational capacity of a GNN model when we have prior knowledge to indicate that some neighboring nodes can be more useful than others. For example, consider the case of knowledge graph reasoning task shown in Figure 1.1c where one has to predict the relation between `S.Curry` and `California`. We can imagine that this prediction depends more on `S.Curry`'s spouse `Ayesha` than `Under Armour`, a brand he endorses. Ideally, the attention mechanism should allow a GNN model to differentiate between such relevant and irrelevant nodes. In fact, as we will show it is the attention mechanism that enables the approach proposed in this thesis, GraIL, to learn path-based rules. While the GAT-style attention computation (Equation (3.6)) is known to work well with graph data, in principle, any standard attention mechanism from the deep learning literature can be used. GraIL uses a variation of attention using MLPs as we will see in Section 4.1.3.

3.1.2 Generalized UPDATE function

In the simple realization of the UPDATE function in Equation (3.4), the node representation is updated with a linear combination of neighborhood message $\mathbf{m}^l_u$ and the node's previous representation $\mathbf{h}^{l-1}_u$. While simple and effective in many settings, this realization has its limitations. These limitations are akin to the problems encountered

by deeper CNNs or RNNs. For one, as in any feed-forward neural network it gets difficult to learn with large number of layers due to the nested non-linear structure that prevents the ease of passing information and gradient along the computational path. For another, the repeated application of matrix multiplication often leads to numerical instabilities (e.g., exploding gradients) in deeper GNNs. This issue is analagous to the numerical instabilities observed for basic RNN models. Lastly, the linear combination of neighboring message with nodes' representation at every layer leads to an issue known as *over-smoothing* [94]. In simpler terms, *over-smoothing* refers to an issue where after a large number of iterations of message passing, the representations for all the nodes in the graph become very similar to one another. Below, we will describe some generalizations to the UPDATE function which alleviate these issues and also provide additional expressive power. We explore all these variations in GraIL.

Skip connections

In order to alleviate the difficulty in learning through non-linearities and large numbers of layers, various forms of skip connections can be employed. This is largely inspired by the ideas of highway connections [97] and residual connections [98] used to effectively train deep CNNs. These skip-connections can be used in conjunction with most other GNN update functions. One form of such skip-connections used in GNNs was proposed by Pham et al.[99]:

$$\text{UPDATE}_{\text{interpolate}}(\mathbf{h}_u^{l-1}, \mathbf{m}_{\mathcal{N}(u)}) = \alpha_1 \odot \text{UPDATE}_{\text{base}}(\mathbf{h}_u^{l-1}, \mathbf{m}_{\mathcal{N}(u)}) + \alpha_2 \odot \mathbf{h}_u^{l-1}, \quad (3.7)$$

where $\text{UPDATE}_{base}(\mathbf{h}_u^{l-1}, \mathbf{m}_{\mathcal{N}(u)})$ denotes the simple UPDATE function defined in Equation (3.4) and $\alpha_1, \alpha_2 \in [0,1]^d$ are gating vectors. This eases the learning of deep GNNs by partially opening the non-linearity gate that lets previous states to propagate through layers. W. L. Hamilton et al.[21] propose a concatenation strategy which is inspired by similar motivations. These concatenation and residual connections are

simple strategies that can help to alleviate the over-smoothing issue in GNNs, while also improving the numerical stability of optimization.

Gated updates

While the AGGREGATE and UPDATE functions of a GNN are generally interpreted as convolution filters applied to all the nodes in the graph, the different layers of GNN can also be interpreted as being analogous to the time steps of a RNN [37]. In particular, the update function of every layer of a GNN can be interpreted as taking in an 'observation' (the neighborhood message) and updating the 'hidden state' (the node representation) of the nodes. This interpretation allows for natural use of various gating mechanisms of designed for RNN architectures. For example, Li et al.[100] show the effectiveness of using a Gated Recurrent Unit (GRU) [101] cell as the UPDATE function:

$$\mathbf{h}_u^l = \text{GRU}(\mathbf{h}_u^{l-1}, \mathbf{m}_{\mathcal{N}(u)}^l). \tag{3.8}$$

In a similar fashion, Selsam et al. [22] use LSTM [102] cell as the UPDATE function. In general, any update function defined for RNN can be used in the context context of GNNs. The hidden state argument of the RNN update function (usually denoted $\mathbf{h}^t$) is replaced with the node's hidden state, and the observation vector (usually denoted $\mathbf{x}^t$) with the message aggregated from the local neighborhood. Generally, when using these gating mechanisms, the UPDATE function is not only shared across different nodes but also shared across message passing layers of the GNN. Just as these update mechanisms help capture long range dependencies in sequences, they help capture long-distance relations in the graphs by allowing for large number of message passing layers in GNNs (e.g., more than 10 layers).

Jumping knowledge connections

So far, we have used the final layer node representations as the node embeddings for any downstream task, i.e.,

$$\mathbf{z}_u = \mathbf{h}_u^L, \forall u \in \mathcal{E}.$$

The limitations of this approach motivated the skip connections and gated updates. An alternate way to alleviate much of the same limitations is to use the node representations from intermediate layers of the message passing iterations. A straight-forward way to do that is to simply concatenate the node representations for all the layers:

$$\mathbf{z}_u = f_{JK}(\mathbf{h}_u^0 \oplus \mathbf{h}_u^1 \oplus \cdots \oplus \mathbf{h}_u^L), \tag{3.9}$$

where $f_{JK}(.)$ is an arbitrary differentiable function. This strategy is known as adding *jumping knowledge (JK) connections* [94]. This not only alleviates the over-smoothing problem by giving access to all the intermediate representations including the initial features, if any, but has the potential to adaptively select individual neighborhood sizes for different nodes. In other words, if the function $f_{JK}(.)$ is made to be node-specific the model has the flexibility to select different neighborhood sizes from 1-hop to L-hop for each node. This can be particularly useful because the *influential neighborhood* for each node is dependent on the node's global position in the graph [94]. For example, a spoke node (i.e., a node of low degree and low-degree neighbors) probably needs information from higher hop neighborhood to get access to important parts of the graph than a hub node needs to. In general, JK-connections have proved to give consistent improvements across a wide variety of tasks and models, including the ones proposed in this thesis.

3.2 GNNs FOR MULTI-RELATIONAL GRAPHS

All the variations of GNNs we have seen so far are designed for simple graphs, i.e., graphs where all the edges are of a single type. All of these methods could naturally

be extended to multi-relational graphs where edges can be directed and of different types. A straight-forward way is to have separate transformation matrices for different edge types (relations) [57]. In particular, the aggregation function in Equation (3.3) can be generalized as

$$\mathbf{m}_{\mathcal{N}(u)}^{l} = \sum_{\tau \in \mathcal{R}} \sum_{v \in \mathcal{N}_{\tau}(u)} \mathbf{W}_{\tau} \mathbf{h}_{v}^{l-1}, \tag{3.10}$$

where $\mathcal{N}_{\tau}(u)$ denotes the neighbors of node u connected by edge of types τ. Schlichtkrull et al.[57] refer to this as the R-GCN approach, and we build upon the R-GCN approach in this thesis. A more subtle distinction of this message passing scheme from the case of simple graphs is the directed nature of edges in multi-relational graphs. In particular, the $\mathcal{N}_{\tau}(u)$ in Equation (3.10) can denote incoming-neighbors (resulting in head-to-tail message passing) or outgoing-neighbors (resulting in tail-to-head message passing) of node u. Messages can also be passed in both directions by augmenting the graph with inverse relations as is done in the originally proposed R-GCN [57]. This aspect of modelling choice is poorly studied in the literature and in most approaches, including the one proposed in this thesis, the design choice is empirically driven.

One of the key challenges of applying this message passing scheme to highly multi-relational graphs is the rapid growth in number of parameters with number of relations in the graph. In practice, this can lead to overfitting on rare relations and to very large models. Schlichtkrull et al.[57] propose two different regularization strategies to mitigate this issue – 1) basis sharing and, 2) block diagonal decomposition. The basis sharing strategy, which proved more effective for relation prediction task, represents each transformation matrix as:

$$\mathbf{W}_{\tau} = \sum_{i=1}^{b} \alpha_{i,\tau} \mathbf{B}_{i}. \tag{3.11}$$

Here, the relation transformation matrices are defined as the linear combination of b basis matrices $\mathbf{B}_1, \ldots, \mathbf{B}_b$ and the only relation specific parameters are the b coefficients $\alpha_{1,\tau}, \ldots, \alpha_{b,\tau}$. The basis matrices are hence shared across all relations thus reducing the number of parameters and regularizing the model.

3.3 GNNs for downstream tasks

The architectures described so far embed each node of the graphs to a low-dimensional embedding. Node-specific tasks like node classification can act on these node embeddings to make predictions [20, 21]. Other graph-level tasks like graph classification require graph embeddings that can be obtained by pooling the embeddings of all the nodes in the graph [21, 95, 103]. A straight-forward pooling mechanism can be a simple addition of all the embeddings but a number of sophisticated and more expressive pooling mechanisms have been proposed [104, 105].

Other than the node and graph classification, the task of interest in this thesis is of relation prediction. As we have seen in the previous chapter, approaches to the relation prediction task can be formalized in a generalized encoder-decoder framework. In that context, all the GNN variants we have seen so far are just more expressive encoders. These GNN-encoders are trained by coupling them with the decoder functions we have seen in Section 2.1.1. R-GCN [57] obtains its best performance when coupled with DistMult decoder [52]. While R-GCN represents the current dominant multi-relational GNN approach, a recent extension [58] addresses some of its shortcomings. In particular, while R-GCN employs a more expressive node encoder the relation encoder is still a shallow embedding look-up. CompGCN [58] addresses that by using GCNs to obtain both node and relation embeddings.

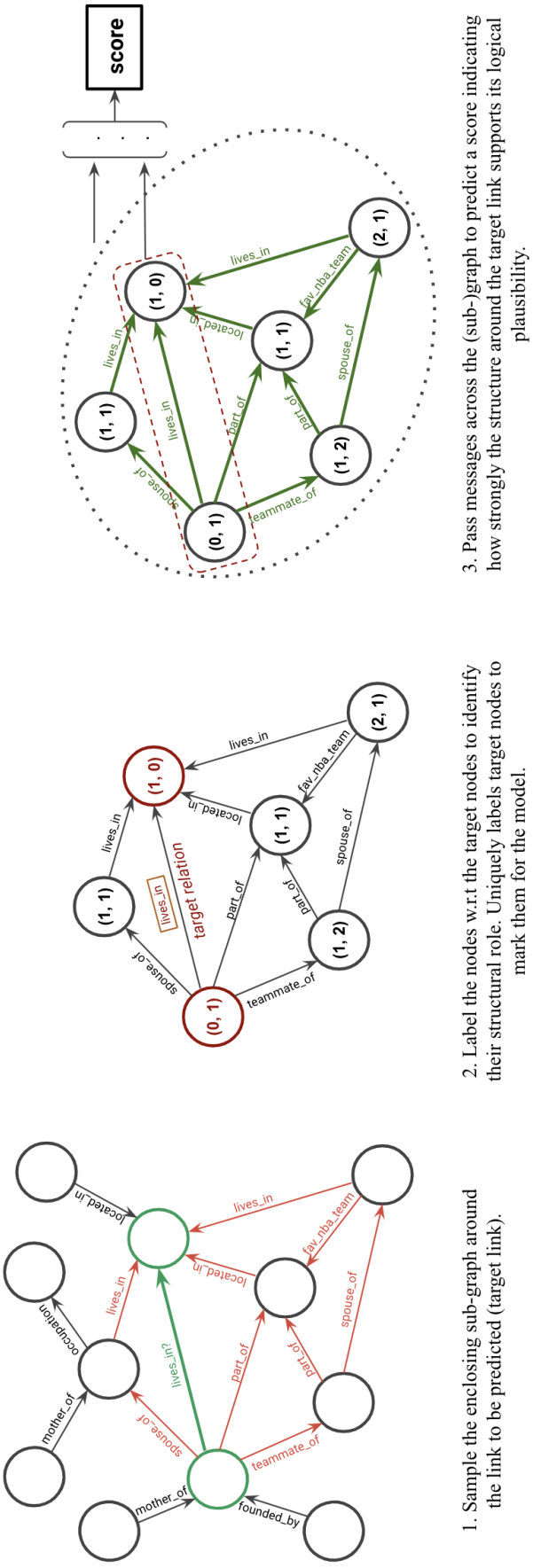
Relation prediction simplifies to link prediction in case of simple graphs. Srinivasan et al.[106] have theoretically shown that tasks like link prediction require pairwise representations rather than individual node representations. This implies that the GNNs of the above kind which aim to embed each node's local neighborhood into an individual embedding are fundamentally limited in their capacity to perform pairwise link/relation prediction. M. Zhang et al.[107] and our method, GraIL, reinforce this finding by proposing and demonstrating the superiority of a method that performs link/relation prediction by learning pairwise structural representations.

4

GraIL

In this chapter, we will describe the proposed model, GraIL(Graph Inductive Learning), in detail. We build up on the the many variations of message passing paradigm of graph neural networks introduced in Chapter 3 and the reference task is of *relation prediction* in knowledge graph as described in Chapter 2. We also provide theoretical motivations for the choices made in our model design and prove that the proposed model can represent a useful subset of logical rules. Towards the end, we also address the computational limitations of our framework and discuss some ways we can address them.

The *relation prediction* task boils down to scoring a triplet (u, r_t, v), i.e., predicting the likelihood of a possible relation r_t between a *head* node u and *tail* node v in a knowledge graph, where we refer to nodes u and v as *target nodes* and to r_t as the *target relation*. The key idea behind our approach is to predict the relation between the two *target nodes* from the subgraph structure around those two *target nodes*. We do not use any node attributes (e.g., textual features) in order to test GraIL's ability to learn and generalize solely from structure. Since it only ever receives structural information (i.e., the subgraph structure and structural node features) as input, the only way GraIL can complete the relation prediction task is to learn the entity-independent relational semantics that underlie the knowledge graph.



1. Sample the enclosing sub-graph around the link to be predicted (target link).

2. Label the nodes w.r.t the target nodes to identify their structural role. Uniquely labels target nodes to mark them for the model.

3. Pass messages across the (sub-)graph to predict a score indicating how strongly the structure around the target link supports its logical plausibility.

Figure 4.1: Visual illustration of GraIL for inductive relation prediction.

4.1 Model Description

Our approach to scoring triplets can be roughly divided into three sub-tasks: (i) extracting the enclosing subgraph around the target nodes, (ii) labeling the nodes in the extracted subgraph, and (iii) scoring the labeled subgraph using a GNN (Figure 4.1).

4.1.1 Step 1: Subgraph Extraction

We assume that local graph neighborhood of a particular triplet in the knowledge graph will contain the logical evidence needed to deduce the relation between the target nodes. In particular, we assume that the paths connecting the two target nodes contain the information that could imply the target relation. For example, in Figure 4.1 the evidence for target relation `lives_in` can be found in the paths `spouse_of` $\wedge$ `lives_in` or `part_of` $\wedge$ `located_in` in the local neighborhood that connect the two target nodes. Hence, as a first step, we extract the enclosing subgraph around the target nodes. We define the *enclosing subgraph* between nodes u and v as the graph induced by all the nodes that occur on a path of a specific length between u and v. We make the following simple observation about nodes on a path between two nodes which motivates our approach to extract enclosing subgraphs.

Observation 1. *In any given graph, let the nodes on a path of length λ between two different nodes u and v constitute the set $P_{uv}(\lambda)$. The maximum distance of any node on such a path, $v \in P_{uv}(\lambda)$, from either u or v is $\lambda - 1$.*

Based on this observation, our intuition to extract the enclosing subgraphs is to get the intersection of neighbors of the two target nodes followed by a pruning procedure. More precisely, let $\mathcal{N}_k(u)$ and $\mathcal{N}_k(v)$ be set of nodes in the k-hop (undirected) neighborhood of the two target nodes in the knowledge graph. We compute the enclosing subgraph by taking the intersection, $\mathcal{N}_k(u) \cap \mathcal{N}_k(v)$, of these k-hop neighborhood sets and then prune nodes that are isolated or at a distance greater than k from either of

the target nodes. Following the Observation 1, this would give us all the nodes that occur on a path of length at most $k+1$ between nodes u and v. A detailed description of this approach is given in Algorithm 1.

Algorithm 1: Enclosing subgraph extraction algorithm

Input : Multi-relational Graph induced by nodes $\mathcal{E}$ and relation types $\mathcal{R}$: $\mathcal{G}(\mathcal{E};\mathcal{R})$; target nodes: (u, v); neighborhood size: k

Output: Set of nodes in the enclosing subgraph induced by k-hop neighborhood around the given target nodes: $\mathcal{G}_{(u,v)} : \{n | n \in P_{uv}(k+1)\}$

$\mathcal{N}_k(u) = \texttt{get\_und\_neighborhood}(u, k; \mathcal{G}(\mathcal{E};\mathcal{R}))$

$\mathcal{N}_k(v) = \texttt{get\_und\_neighborhood}(v, k; \mathcal{G}(\mathcal{E};\mathcal{R}))$

$\mathrm{c}_n = \mathcal{N}_k(u) \cap \mathcal{N}_k(v)$

$\mathcal{G}_{(u,v)} = \{n \in \mathrm{c}_n | \mathtt{d}(n, u; \mathcal{G}(\mathcal{E} \setminus \{v\}, \mathcal{R})) \leq k \text{ and } \mathtt{d}(n, v; \mathcal{G}(\mathcal{E} \setminus \{u\}), \mathcal{R}) \leq k\}$

Here, the $\texttt{get\_und\_neighborhood}(u, k; \mathcal{G}(\mathcal{E};\mathcal{R}))$ function returns the set of nodes in k-hop undirected neighborhood of node u in the graph $\mathcal{G}(\mathcal{E};\mathcal{R})$, $\mathtt{d}(x, y; \mathcal{G}(\mathcal{E}, \mathcal{R}))$ gives the shortest distance between nodes x and y in graph. Note that while extracting the enclosing subgraph we completely ignore the direction of the edges. However, the direction is preserved while passing messages with a graph neural network, a point revisited later. Also, note that the target tuple/edge (u, r_t, v) is added to the extracted subgraph to enable message passing between the two target nodes.

4.1.2 Step 2: Node labeling

After extracting the *enclosing subgraph* around the target nodes, the goal is to score the subgraph using a graph neural network. As noted in Chapter 3, GNNs require a node feature matrix, $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, to initialize the node representations before the first layer of message passing. This is one of the key advantages the message passing formalism of GNN provides—the natural ability to leverage information from node features. However, in our approach we defer from using any node features to study the predictive abilities of our model to reason solely structural information.

Instead of using external node features or attributes, we extend the double-radius node labelling scheme [107] that reinforces the topological structure of the subgraph. In particular, for each node in the enclosing subgraph around target nodes u and v we assign the following features to each node

$$\mathbf{X}_i = [\texttt{one-hot}(\texttt{d}(i, u, \mathcal{G}_s(\mathcal{E} \setminus \{v\}))) \oplus \texttt{one-hot}(\texttt{d}(i, v, \mathcal{G}_s(\mathcal{E} \setminus \{u\})))] \quad \forall i \in \mathcal{G}_{(u,v)} \quad (4.1)$$

where `d(.)` is the same shortest distance function described in Algorithm 1, `one-hot(.)` converts a scalar into the corresponding one-hot vector representation, and $\oplus$ denotes concatenation of two vectors. In other words, each node, i, in the subgraph around target nodes u and v is labeled with the tuple $(d(i, u), d(i, v))$, where $d(i, u)$ denotes the shortest distance between nodes i and u without counting any path through v (likewise for $d(i, v)$).

This node labelling scheme captures the topological position of each node with respect to the target nodes and reflects its structural role in the subgraph. Moreover, it is important for the model to identify the target nodes in the sub-graph to be able to predict a relation between them. This labelling scheme sets the two target nodes as the anchor points of the subgraph and helps the model clearly identify them. In particular, the target nodes, u and v, are uniquely labeled $(0, 1)$ and $(1, 0)$. Note that as a consequence of Observation 1, the dimension of node features constructed this way is bounded by the number of hops considered while extracting the enclosing subgraph. In particular, when considering k-hop enclosing subgraph, the maximum dimension of the node features, by definition (Equation 4.1), is $2k$. This makes our model extremely parameter efficient, as we will detail in Section 5.

4.1.3 Step 3: GNN scoring

The final step in our framework is to use a GNN to score the likelihood of tuple (u, r_t, v) given $\mathcal{G}_{(u,v)}$—the extracted and labeled subgraph around the target nodes. We adopt the general message-passing scheme described in Section 3.1 where a node

representation is iteratively updated by combining it with aggregation of it's neighbors' representation. In particular, the k^{th} layer of our GNN is given by,

$$\mathbf{m}^k_{\mathcal{N}(t)} = \text{AGGREGATE}^k\left(\left\{\mathbf{h}^{k-1}_s : s \in \mathcal{N}(t)\right\}\right), \tag{4.2}$$

$$\mathbf{h}^k_t = \text{UPDATE}^k\left(\mathbf{h}^{k-1}_t, \mathbf{m}^k_{\mathcal{N}(t)}\right), \tag{4.3}$$

where $\mathbf{a}^k_t$ is the aggregated message from the neighbors, $\mathbf{h}^k_t$ denotes the latent representation of node t in the k-th layer, and $\mathcal{N}(t)$ denotes the set of immediate neighbors of node t. The initial latent node representation of any node i, $\mathbf{h}^0_i$, is initialized to the node features, $\mathbf{X}_i$, built according to the labeling scheme given by Equation 4.1. This framework gives the flexibility to plug in different AGGREGATE and UPDATE functions resulting in various GNN architectures (Sections 3.1.1 and 3.1.2).

Inspired by the multi-relational R-GCN [57] and edge attention [96], we define our AGGREGATE function as

$$\mathbf{m}^k_{\mathcal{N}(t)} = \sum_{r=1}^{R} \sum_{s \in \mathcal{N}_r(t)} \alpha^k_{rr_tst} \mathbf{W}^k_r \mathbf{h}^{k-1}_s,$$

where R is the total number of relations present in the knowledge graph; $\mathcal{N}_r(t)$ denotes the immediate *outgoing* neighbors of node t under relation r; $\mathbf{W}^k_r$ is the trainable transformation matrix used to propagate messages in the k-th layer over relation r; $\alpha^k_{rr_tst}$ is the edge attention weight at the k-th layer corresponding to the edge connecting nodes s and t via relation r. This attention weight, a function of the source node t, neighbor node s, edge type r and the target relation to be predicted r_t, is given by

$$\mathbf{s} = \text{ReLU}\left(\mathbf{A}^k_1[\mathbf{h}^{k-1}_s \oplus \mathbf{h}^{k-1}_t \oplus \mathbf{e}^a_r \oplus \mathbf{e}^a_{r_t}] + \mathbf{b}^k_1\right)$$

$$\alpha^k_{rr_tst} = \sigma\left(\mathbf{A}^k_2\mathbf{s} + \mathbf{b}^k_2\right).$$

Here, $\mathbf{h}^k_s$ and $\mathbf{h}^k_t$ denote the latent node representation of respective nodes at k-th layer of the GNN, $\mathbf{e}^a_r$ and $\mathbf{e}^a_{r_t}$ denote learned attention embeddings of respective relations, and $\mathbf{A}_i$ and $\mathbf{b}_i$ are trainable matrices. Note that the attention weights are

not normalized and instead come out of a sigmoid gate that regulates the information aggregated from each neighbor. As a regularization measure, we adopt the basis sharing mechanism, introduced by Schlichtkrull et al.[57] and described in Section 3.2, among the transformation matrices of each layer, $\mathbf{W}_r^k$. We also implement a form of *edge dropout*, where edges are randomly dropped from the graph while aggregating information from the neighborhood. This proved to be an effective regularization strategy.

Our UPDATE function is defined as the weighted linear combination of the neighboring message and the current node representation:

$$\mathbf{h}_t^k = \text{ReLU}\left(\mathbf{W}_{self}^k \mathbf{h}_t^{k-1} + \mathbf{m}_{\mathcal{N}(t)}^k\right). \tag{4.4}$$

With the AGGRERGATE and UPDATE functions defined, we obtain the node representations after L layers of message passing. The final node embeddings are given by the concatenation of node representations at all layers,

$$\mathbf{z}_i = [\mathbf{h}_i^1 \oplus \cdots \oplus \mathbf{h}_i^L] \ \forall i \in \mathcal{G}_{(u,v)}.$$

This is inspired by the JK-connection mechanism [94] described in Section 3.1.2, which allows for flexible neighborhood ranges for each node. Empirically, the addition of these JK-connections made our model's performance robust to the number of layers of the GNN.

A subgraph representation of $\mathcal{G}_{(u,v)}$ is then obtained by average-pooling of all the latent node representations:

$$\mathbf{h}_{\mathcal{G}_{(u,v)}}^L = \frac{1}{|\mathcal{G}_{(u,v)}|} \sum_{i \in \mathcal{G}_{(u,v)}} \mathbf{h}_i^L, \tag{4.5}$$

where $\mathcal{G}_{(u,v)}$ denotes the set of vertices in graph $\mathcal{G}_{(u,v)}$.

Finally, to obtain the score for the likelihood of a triplet (u, r_t, v), we create a latent representation of the triplet by concatenating the following vectors

- target nodes' latent representations, $\mathbf{h}_u^L$ and $\mathbf{h}_v^L$: these encode the target nodes neighborhood information.

- subgraph representation, $\mathbf{h}^L_{\mathcal{G}_{(u,v)}}$: summarized information from the subgraph.
- a learned embedding of the target relation, $\mathbf{e}_{r_t}$: the subgraph structure around the target nodes, $\mathcal{G}_{(u,v)}$, is independent of the target relation, r_t; this is the target relation specific input that the model needs to score the triplet (u, r_t, v).

We then pass these concatenated representations through a linear layer, with trainable linear transformation weight matrix $\mathbf{W}$,

$$\text{score}(u, r_t, v) = \mathbf{W}^T[\mathbf{h}^L_u \oplus \mathbf{h}^L_v \oplus \mathbf{h}^L_{\mathcal{G}_{(u,v)}} \oplus \mathbf{e}_{r_t}]. \tag{4.6}$$

For simplicity, we consider the same dimension for all our latent dimensions (e.g., for all hidden layers of GNN, and relation embeddings, $\mathbf{e}_{r_t}$) The AGGREGATE and UPDATE functions can be further generalized as described in Sections 3.1.1 and 3.1.2.

4.1.4 Training Regime

Following the standard and successful practice, we train the model to score positive triplets higher than the negative triplets using a noise-contrastive hinge loss [15]. This follows the *max-margin loss function* introduced in Equation (2.11). To re-iterate, we use the following loss function to train our model via stochastic gradient descent:

$$\mathcal{L} = \sum_{i=1}^{|\mathcal{T}|} \max(0, -\text{score}(p_i) + \text{score}(n_i) + \gamma), \tag{4.7}$$

where $\mathcal{T}$ is the set of all edges/triplets in the training graph; p_i and n_i denote the positive and negative triplets respectively; γ is the margin hyperparameter. In our approach the negative triplet, n_i, is obtained by following the common strategy described in Section 2.1.2 of replacing the head (or tail) of the triplet with a uniformly sampled random entity while filtering out the occasional false negatives.

4.2 Theoretical Analysis

We can show that the GraIL architecture is capable of encoding the same class of *path-based* logical rules that are used in popular rule induction models, such as RuleN [29] and NeuralLP [33] and studied in recent work on logical reasoning using neural networks [108]. For the sake of exposition, we equate edges (u, r, v) in the knowledge graph with binary logical predicates $r(u, v)$ where an edge (u, r, v) exists in the graph iff $r(u, v) = true$.

Theorem 1. *Let $\mathcal{R}$ be any logical rule (i.e., clause) on binary predicates of the form:*

$$r_t(X, Y) \leftarrow r_1(X, Z_1) \wedge r_2(Z_1, Z_2) \wedge ... \wedge r_k(Z_{k-1}, Y),$$

where $r_t, r_1, \ldots, r_k$ are (not necessarily unique) relations in the knowledge graph, $X, Z_1, \ldots, Z_{k-1}, Y$ are free variables that can be bound by arbitrary unique entities. For any such $\mathcal{R}$ there exists a parameter setting Θ for a GraIL model with k GNN layers and where the dimension of all latent embeddings are $d = 1$ such that

$$score(u, r_t, v) \neq 0$$

if and only if $\exists Z_1, ..., Z_{k-1}$ where the body of $\mathcal{R}$ is satisfied with $X = u$ and $Y = v$.

Theorem 1 states that any logical rule corresponding to a path in the knowledge graph can be encoded by the model. GraIL will output a non-zero value if and only if the body of this logical rule evaluates to true when grounded on a particular set of query entities $X = u$ and $Y = v$. The full proof of Theorem 1 is detailed in Section 4.2.1, but the key idea is as follows: Using the edge attention weights it is possible to set the model parameters so that the hidden embedding for a node is non-zero after one round of message passing (i.e., $\mathbf{h}_s^i \neq 0$) if and only if the node s has at least one neighbor by a relation r_i. In other words, the edge attention mechanism allows the model to indicate whether a particular relation is incident to a particular entity, and—since we have uniquely labeled the targets nodes u and v—we can use

this relation indicating property to detect the existence of a particular path between nodes u and v.

We can extend Theorem 1 in a straightforward manner to also show the following:

Corollary 1. *Let $\mathcal{R}_1 ..., \mathcal{R}_m$ be a set of logical rules with the same structure as in Theorem 1 where each rule has the same head $r_t(X, Y)$. Let*

$$\beta = |\{\mathcal{R}_i \ : \exists Z_1, ..., Z_{k-1} \text{ where } \mathcal{R}_i = true \text{ with } X = u \text{ and } Y = v\}|.$$

Then there exists a parameter setting for GraIL with the same assumptions as Theorem 1 such that

$$score(u, r_t, v) \propto \beta.$$

This corollary shows that given a set of logical rules that implicate the same target relation, GraIL can count how many of these rules are satisfied for a particular set of query entities u and v. In other words, similar to rule-induction models such as RuleN, GraIL can combine evidence from multiple rules to make a prediction.

Interestingly, Theorem 1 and Corollary 1 indicate that GraIL can learn logical rules using only one-dimensional embeddings of entities and relations, which dovetails with our experience that GraIL's performance is reasonably stable for dimensions in the range $d = 1, ..., 64$. However, the above analysis only corresponds to a fixed class of logical rules, and we expect that GraIL can benefit from a larger latent dimensionality to learn different kinds of logical rules and more complex compositions of these rules.

4.2.1 Proof of Theorem 1

We prove this Theorem by first proving the following two lemmas.

Lemma 1. *Given a logical rule $\mathcal{R}$ as in 1, we have r_t in the head and r_i is any relation in the body at a distance i from the head. Then the attention weight between*

any nodes, s and t, connected via relation r, $\alpha^l_{rr_tst}$, at layer l can be learnt such that

$$\alpha^l_{rr_tst} > 0$$

if and only if $r = r_l$.

Proof. For simplicity, let us assume a simpler version of $\alpha^l_{rr_tst}$ as follows

$$\alpha^l_{rr_tst} = \text{MLP}(r, r_t).$$

When r and r_t are 1-dimensional scalars (as we assume in Theorem 1), to prove the stated lemma we need the MLP to learn a decision boundary between the true pair $\mathcal{S}^l : \{(r_l, r_t)\}$ and the induced set of false pairs $\bar{S}^l : \{(r_i, r_j)\ \forall (r_i, r_j) \notin \mathcal{S}^l\}$. We also have the flexibility of learning appropriate embeddings of the relations in 1-dimensional space.

This is possible to an arbitrary degree of precision given that MLP with non-linear activation, as is our case, is a universal function approximator [109].

Lemma 2. *For a given rule $\mathcal{R}$ as in 1 which holds true for a pair of nodes, $X = u$ and $Y = v$, it is possible to learn a set of parameters for a GraIL model such that*

$$\mathbf{h}^l_t > 0$$

if and only if node t is connected to node u by a path,

$$r_1(u, Z_1) \wedge r_2(Z_1, Z_2) \wedge ... \wedge r_k(Z_{l-1}, t),$$

of length l.

Proof. The overall message passing scheme of best performing GraIL model is given by

$$\mathbf{h}^l_t = \text{ReLU}\left(\mathbf{W}^l_{self}\mathbf{h}^{l-1}_t + \sum_{r=1}^{R}\sum_{s\in\mathcal{N}_r(t)} \alpha^l_{rr_tst}\mathbf{W}^l_r\mathbf{h}^{l-1}_s\right) \tag{4.8}$$

Without loss of generality [1], we assume all the nodes are labeled with 0, except the node, u, which is labeled 1. Under this node label assignment, for any node t, at a distance d from the node u, $\mathbf{h}_t^l = 0 \ \forall l < d$.

With no loss of generality, also assume $W_r^k = 1, W_{self}^k = 1 \ \forall k, r$. With these assumptions, Equation (4.8) simplifies to

$$\mathbf{h}_t^l = \text{ReLU}\left(\sum_{r=1}^{R} \sum_{s \in \mathcal{N}_r(t)} \alpha_{rr_tst}^l \mathbf{h}_s^{l-1}\right). \tag{4.9}$$

We will now prove our Lemma using induction.

Base case. We will first prove the base case for $l = 1$, i.e., $\mathbf{h}_t^1 > 0$ if and only if t is connected to u via path $r_1(u, t)$

From Equation 4.9, we have that

$$\mathbf{h}_t^1 = \text{ReLU}\left(\sum_{r=1}^{R} \sum_{s \in \mathcal{N}_r(t)} \alpha_{rr_t}^1 \mathbf{h}_s^0\right).$$

According to our simplified node labeling scheme $\mathbf{h}_s^0 \neq 0$ only if $s = u$. And by Lemma 1, $\alpha_{rr_tst}^1 > 0$ only if $r = r_1$. Hence, t must be connected to u via relation r_1 for $\mathbf{h}_t^1$ to be non-zero.

Induction step. Assume the induction hypothesis is true for some λ, i.e., $\mathbf{h}_t^\lambda > 0$ if and only if t is connected to source u by a path $r_1(u, Z_1) \wedge ... \wedge r_\lambda(Z_{\lambda-1}, t)$.

From Equation 4.9 we have that $\mathbf{h}_t^{\lambda+1} > 0$ when the following two conditions are simultaneously satisfied.

1. $\mathbf{h}_s^\lambda > 0$ for some s
2. $\alpha_{rr_tst}^{\lambda+1} > 0$ for some r

As a consequence of our induction hypothesis, Condition 1 directly implies that node s should be connected to source node u by a path $r_1(u, Z_1) \wedge ... \wedge r_\lambda(Z_{\lambda-1}, s)$.

By Lemma 1, Condition 2 implies that $r = r_{\lambda+1}$. This means that node t is connected to node s via relation $r_{\lambda+1}$.

[1] There is no loss of generality because this is a strict simplification of our original labelling scheme without any additional properties.

The above two arguments directly imply that $\mathbf{h}_t^{\lambda+1} > 0$ if and only if node t is connected to source node by a path $r_1(u, Z_1) \wedge ... \wedge r_\lambda(Z_{\lambda-1}, s) \wedge r_{\lambda+1}(s, t)$.

Hence, assuming the lemma holds true for λ, we proved that holds it true for $\lambda+1$. Thus, Lemma 1 is proved by induction.

Proof of Theorem 1. This is a direct consequence of Lemma 2. In particular, without any loss of generality we simplify the final scoring of GraIL to directly be the embedding of the target node v at the last layer k, i.e,

$$\text{score}(u, r_t, v) = \mathbf{h}_v^k$$

According to Lemma 2, $\mathbf{h}_v^k$ is non-zero only when v is connected to u by the body of rule $\mathcal{R}$, hence proving the above stated theorem.

4.3 Computational Complexity and Scalability

Unlike traditional embedding-based approaches, scoring a triplet in the GraIL model requires extracting and processing a subgraph around the candidate edge (u, r_t, v) and running multiple rounds of message passing on this extracted subgraph. Given that our processing includes a node-labelling procedure that requires evaluating shortest paths from the target nodes to all other nodes in the extracted subgraph, the time complexity of GraIL to score a candidate edge (u, r_t, v) is given by

$$O(\log(|\mathcal{E}|)|\mathcal{T}| + |\mathcal{R}|dk),$$

where $|\mathcal{E}|$, $|\mathcal{R}|$, and $|\mathcal{T}|$ are the number of nodes, relation types and triplets, respectively, in the enclosing subgraph induced by u and v, $\mathcal{G}_{(u,v)}$. d is the dimension of the node/relation embeddings and k is the number of GNN layers. Thus, the computational cost of scoring a triplet with GraIL depends largely on the size of the extracted

subgraphs, and the runtime in practice is usually dominated by running Dijkstra's algorithm on these subgraphs.

During training this scoring needs to be done for all the positive and randomly sampled negative edges of the training graph. This quickly gets prohibitively expensive for large graphs. However, note that the procedure of extracting and labelling subgraphs is independent for each training edge. This allows for massive parallelization of the subgraph processing. In particular, we can easily distribute the subgraph extraction and node-labelling procedures across multiple processes. Further, by passing messages solely on the subgraph enclosing a pair of nodes, GraIL avoids the challenges of memory and parallelizability associated with using GNNs on large graphs. In particular, since the extracted subgraphs can be batched, GraIL is naturally suited to multi-GPU training. This computational expense becomes furthermore prohibitive while adopting certain evaluation metrics. We address this issue in the next chapter and discuss some ways to mitigate this in Chapter 6.

5

Experimental results

In this chapter we present a thorough empirical analysis of the proposed model, GraIL. We verify many of the hypotheses we proposed while designing the approach. In particular, we perform experiments to study the following aspects:

1. **Inductive relation prediction**. One of the key motivations for our approach is to learn entity-independent relational semantics. In Theorem 1, we also prove that GraIL can encode inductive logical rules. In this set of experiments we introduce the *inductive relation prediction* task, propose new benchmark datasets, and evaluate GraIL in comparison to existing statistical and differentiable methods, which explicitly do rule induction.

2. **Transductive relation prediction**. The inductive models have a strong structural inductive bias that, we hypothesize, is complementary to existing state-of-the-art transductive knowledge graph embedding methods. In this set of experiments we explore several ensembling strategies and demonstrate that this complementary inductive bias gives significant improvements over the existing state-of-the-art KGE methods in the traditional transductive setting.

3. **Ablation study**. We perform systematic ablation and sensitivity studies to highlight the importance of various components of our proposed framework. For example, we test the practical importance of attention and the node-labeling

scheme that Theorem 1 relies on. Further, we test the sensitivity of our models' performance with respect to the embedding dimension and the neighborhood size of the extracted subgraphs and reinforce some of our assumptions about the models' behaviour.

Datasets

We perform all our experiments on three benchmark knowledge completion datasets (and other variants derived from them).

WN18RR. This is a subset derived from the WordNet (WN) dataset originally introduced by Bordes et al.[42]. WordNet is carefully curated knowledge repository of english words and their meanings. Its entities correspond to word senses, and relationships define lexical relations between them. WN18RR [17] corresponds to a subset which contains only triplets with the 18 most frequent relations. In addition to that, it also filters out easy inverse relations to avoid any leakage into the test set.

FB15k-237. This is derived from the FreeBase dataset originally proposed by Bordes et al.[15]. FreeBase is a collection of general facts like `(Ottawa, capitol_of, Canada)`. FB15k-237 [110] is again a subset of the original dataset consisting of only the most frequent 237 relations and the inverse relations removed. As opposed to WN18RR, which was constructed in a more or less controlled environment, FB15k-237 is collected by crowd-sourcing and is in general more prone to noisy data.

NELL-995. As presented by Xiong et al.[111], this is a subset of 995th iteration of the knowledge graph collected by Never-Ending Language Learning (NELL) system [43]. NELL is an automated natural language processing system that continuously collects fact by crawling the web. We remove certain relations from NELL-995 (e.g., latitude and longitude information of places), which do not contribute to the logical deduction of missing relations. NELL-995 is mid-way between controlled and accurate knowledge graph (WN18RR), and crowd-sourced noisy knowledge graph (FB15k-237).

Evaluation regime

Across all our experiments the fundamental task we test the models on is the *relation prediction* task. As described in Section 2.1.3, there are two widely followed evaluation protocols for *relation prediction*:

- **Classification metric**. This formulation asks the model to score the correctness of a given triplet (u, r, v). We use the area under the precision-recall curve (AUC-PR) to reflect the classification abilities of the models. In particular, we sample an equal number of negative triplets as the positive triplets following the simple and standard practice of replacing the head (or tail) with a random entity. We then score both the positive and the negative triplets to compute the AUC-PR score of the respective models.

- **Ranking metric**. In this formulation, for a given triplet $(u, r, ?)$ or $(?, r, v)$ we ask the model to rank a set of candidate entities. The rank of the correct entity (u or v) among all of the candidate set entities reflects the reasoning abilities of the model. We report Hits@k (for k = 1, 5, and 10) and Mean Reciprocal Rank (MRR) as the summary ranking metrics. Note that these ranking metrics are very sensitive to not only the nature of the entities in the candidate set but also the size of the candidate set. While the general strategy is to consider the entire set of entities in the knowledge graph as the candidate set, due to the computational cost of our method we adopt an approximation where we sample a random set of 50 candidate entities to make the candidate set. We argue that this is a reasonable approximation since in practice smaller and more relevant relation-specific candidate sets can be derived using type constraints and metadata available [66].

	Train graph ($\mathcal{G}_{\text{tr}}$)
Train edges	`(LeBron, lives_in, Ohio)`, `(Savannah, lives_in, Ohio)`, `(LeBron, spouse_of, Savannah)`, `(Britney, lives_in, LA)`, `(A. Davis, spouse_of, Britney)`
Test edges	`(A. Davis, lines_in, ?)`
	Inductive test graph ($\mathcal{G}_{\text{ind}}$)
Train edges	`(S. Curry, spouse_of, Ayesha)`, `(Ayesha, lives_in, CA)`
Test edges	`(S. Curry, lives_in, ?)`

Table 5.1: Illustration of the inductive dataset splits

5.1 Inductive Relation Prediction

As illustrated in Figure 1.1c and here in Table 5.1, an inductive setting evaluates a models' ability to generalize to unseen entities. In a fully inductive setting the sets of entities seen during training and testing are disjoint. More generally, the number of unseen entities during testing can be varied from only a few new entities being introduced to a fully-inductive setting (Figure 1.1c). The proposed framework, GraIL, is invariant to the node identities so long as the underlying semantics of the relations (i.e., the schema of the knowledge graph) remains the same. We demonstrate our inductive results in the extreme case of having an entirely new test graph with a new set of entities.

5.1.1 Inductive Benchmark Datasets

The WN18RR, FB15k-237, and NELL-995 benchmark datasets were originally developed for the transductive setting, i.e., the entities of the standard test splits are a subset of the entities in the training splits (Figure1.1b). In order to facilitate inductive testing, we create new fully-inductive benchmark datasets by sampling disjoint subgraphs from the knowledge graphs in these datasets.

In particular, each of our datasets consist of a pair of graphs: *train graph* ($\mathcal{G}_{\text{tr}}$) and *inductive test graph* ($\mathcal{G}_{\text{ind}}$). These two graphs (i) have disjoint set of entities and (ii)

$\mathcal{G}_{\text{tr}}$ contains all the relations present in $\mathcal{G}_{\text{ind}}$. The overall algorithm used to generate fully-inductive benchmark datasets is presented in Algorithm 2. To generate the *train graph*, $\mathcal{G}_{\text{tr}}$, we sampled several entities uniformly to serve as roots and then take the union of the k-hop neighborhoods surrounding the roots (Step 1 and 2). We get the subset of relation types, r_t, that exist among the nodes of $\mathcal{G}_{\text{tr}}$ (Step 3). We remove the sampled training graph from the whole graph and sample the inductive test graph, $\mathcal{G}_{\text{ind}}$, using the same procedure except we only consider edges of type $r \in r_t$ (Step 4 and 5). For robust evaluation, the parameters of Algorithm 2 are adjusted to obtain four different pairs of $\mathcal{G}_{\text{tr}}$ and $\mathcal{G}_{\text{ind}}$ with increasing number of nodes and edges for each benchmark knowledge graph. The statistics of these inductive benchmarks is provided in Table 5.2.

Algorithm 2: Inductive knowledge graph generation algorithm

Input : Multi-relational Graph induced by nodes $\mathcal{E}$ and relation types $\mathcal{R}$: $\mathcal{G}(\mathcal{E}; \mathcal{R})$; number of seed nodes for training graph: n_t; number of seed nodes for inductive test graph: n_i; neighborhood size: k

Output: Train graph: $\mathcal{G}_{\text{tr}}$, Inductive test graph: $\mathcal{G}_{\text{ind}}$

1 $\mathcal{S}_t = \texttt{uniform\_sample}(\mathcal{E}, n_t)$
2 $\mathcal{D} = \bigcup_{u_j \in \mathcal{S}_t} \texttt{get\_und\_neighborhood}(u_j, k; \mathcal{G}(\mathcal{E}; \mathcal{R}))$
3 $\mathcal{G}_{\text{tr}} = \mathcal{G}(\mathcal{D}; \mathcal{R})$
4 $\text{r}_\text{t} = \texttt{get\_rels}(\mathcal{G}_{\text{tr}})$
5 $\mathcal{S}_i = \texttt{uniform\_sample}(\mathcal{E} \setminus \mathcal{S}_t, n_i)$
6 $\mathcal{I} = \bigcup_{u_j \in \mathcal{S}_i} \texttt{get\_und\_neighborhood}(u_j, k; \mathcal{G}(\mathcal{E} \setminus \mathcal{S}_t; \text{r}_\text{t}))$
7 $\mathcal{G}_{\text{ind}} = \mathcal{G}(\mathcal{I}; r_t)$

In the inductive setting, a model is trained on $\mathcal{G}_{\text{tr}}$ and tested on $\mathcal{G}_{\text{ind}}$. We randomly select 10% of the edges/tuples in the *inductive test graph* as test edges. Consider the expository example presented in Table 5.1. An inductive model is trained on the train edges of $\mathcal{G}_{\text{tr}}$ to ideally learn the rule,

$$\exists Y.(X, \texttt{spouse\_of}, Y) \wedge (Y, \texttt{lives\_in}, Z) \rightarrow (X, \texttt{lives\_in}, Z),$$

and then evaluated on $\mathcal{G}_{\text{ind}}$, which has its own train and test edges with an entirely new set of entities. Having learnt the underlying relational semantics, the model is

Table 5.2: Statistics of inductive benchmark datasets

		WN18RR			FB15k-237			NELL-995		
		rel	nodes	links	rel	nodes	links	rel	nodes	links
v1	train	9	2746	6678	183	2000	5226	14	10915	5540
	ind-test	9	922	1991	146	1500	2404	14	225	1034
v2	train	10	6954	18968	203	3000	12085	88	2564	10109
	ind-test	10	2923	4863	176	2000	5092	79	4937	5521
v3	train	11	12078	32150	218	4000	22394	142	4647	20117
	ind-test	11	5084	7470	187	3000	9137	122	4921	9668
v4	train	9	3861	9842	222	5000	33916	77	2092	9289
	ind-test	9	7208	15157	204	3500	14554	61	3294	8520

expected to make the right prediction on the test edges based on the train edges of $\mathcal{G}_{\text{ind}}$. Contrasting this, in the *transductive setting*, models are trained on the train edges of $\mathcal{G}_{\text{tr}}$ and evaluated on the test edges from the same graph $\mathcal{G}_{\text{tr}}$.

5.1.2 Baselines

We compare GraIL with two other end-to-end differentiable methods, Neural-LP [33] and DRUM [34]. To the best of our knowledge, these are the only differentiable methods capable of inductive relation prediction. We also compare against a state-of-the-art statistical rule mining method, RuleN [29], which performs competitively with embedding-based methods in the transductive setting. RuleN represents the current state-of-the-art in inductive relation prediction on knowledge graphs.

Neural LP and DRUM. These are the first differentiable methods proposed to learn logical rules for knowledge graph reasoning. Neural-LP [33] represents the graph structure by constructing TensorLog [112] operators per relation using a portion of the knowledge graph. These TensorLog operators are further chained to compute a score for each triplet, and rules are learned by maximizing this score. Sadeghian et al.[34] recently proposed a very similar approach but alleviated some of the theoretical limitations of the previous approaches. In particular, they show that Neural-LP

[33] learns at least two highly confident inaccurate rules for every correct rule. This limitation is overcome by generalizing the previous approach with a low-rank tensor approximation. To the best of our knowledge, these two methods are the only differentiable methods capable of inductive relation prediction.

RuleN. We also compare against a state-of-the-art statistical rule mining method, RuleN [29], which performs competitively with embedding-based methods in the transductive setting. RuleN represents the current state-of-the-art in inductive relation prediction on knowledge graphs. It explicitly extracts path-based rules of the kind as shown in Equation (1.1).

5.1.3 Hyperparameter settings

For both Neural-LP and DRUM, we use the implementations publicly provided by the authors with their best configurations[1,2]. Using the original terminology of RuleN [29], we train it to learn rules from paths of length up to 4[3]. By Observation 1 nodes in paths of length 4 appear in the 3-hop neighborhood around the target nodes. In order to maintain a fair comparison, we sample 3-hop enclosing subgraphs around the target links for our GNN approach. We employ a 3-layer GNN with the dimension of all latent embeddings equal to 32. The basis dimension for basis decomposition (denoted by b in equation (3.11)) is set to 4 and the edge dropout rate to 0.5.[4] Experiments were run for 50 epochs on a GTX 1080 Ti with 12 GB RAM. The Adam optimizer was used with a learning rate of 0.01, L2 penalty of 5e-4, and default values for other parameters. The margin in the loss was set to 10. Gradient were clipped at a norm of 1000.[5] The model was evaluated on the validation edges of training graph ($\mathcal{G}_{\text{tr}}$) and saved every three epochs with the best performing checkpoint used for testing on test

[1] https://github.com/fanyangxyz/Neural-LP
[2] https://github.com/alisadeghian/DRUM
[3] http://web.informatik.uni-mannheim.de/RuleN/
[4] These parameters were selected after sweeping over a range of other values and picking the value that gives the best performance on the validation set.
[5] These values were chosen as the default for these parameters.

Table 5.3: Inductive results on datasets derived from WN18RR. N denotes Neural-LP, D denotes DRUM, R denotes RuleN, G denotes GraIL

(a) AUC-PR

	v1	v2	v3	v4
N	86.02	83.78	62.90	82.06
D	86.02	84.05	63.20	82.06
R	90.26	89.01	76.46	85.75
G	**94.32**	**94.18**	**85.80**	**92.72**

(b) Hits@10

	v1	v2	v3	v4
N	74.37	68.93	46.18	67.13
D	74.37	68.93	46.18	67.13
R	81.91	78.23	53.39	71.59
G	82.45	78.68	58.43	73.41

(c) Hits@5

	v1	v2	v3	v4
N	74.37	68.93	45.92	67.13
D	74.37	68.93	46.05	67.13
R	81.91	78.23	53.22	71.59
G	82.45	78.68	57.19	73.41

(d) Hits@1

	v1	v2	v3	v4
N	68.34	66.89	41.16	65.84
D	69.60	67.46	42.17	66.11
R	76.06	76.53	48.60	70.57
G	78.19	76.30	50.33	72.39

(e) MRR

	v1	v2	v3	v4
N	71.74	68.54	44.23	67.14
D	72.46	68.82	44.96	67.27
R	79.15	77.82	51.53	71.65
G	80.45	78.13	54.11	73.84

edges of induction test graph ($\mathcal{G}_{\text{ind}}$). The code for reproducing the results reported is made publicly available[6].

5.1.4 Experimental Results

Tables 5.3, 5.4e, and 5.5 report the mean classification metrics (AUC-PR) and a range of ranking metrics (MRR, Hits@10, Hits@5, Hits@1) averaged over 5 runs of randomly sample negative samples [7]. These results are reported over all the inductive datasets mentioned in Table 5.2 for the four models described above.

As we can see, GraIL significantly outperforms the inductive baselines across all

[6]https://github.com/kkteru/grail

[7]The variance was very low in all the settings, so the standard errors are omitted in these tables.

Table 5.4: Inductive results on datasets derived from FB15k-237. N denotes Neural-LP, D denotes DRUM, R denotes RuleN, G denotes GraIL

(a) AUC-PR

	v1	v2	v3	v4
N	69.64	76.55	73.95	75.74
D	69.71	76.44	74.03	76.20
R	75.24	88.70	91.24	91.79
G	**84.69**	**90.57**	**91.68**	**94.46**

(b) Hits@10

	v1	v2	v3	v4
N	52.92	58.94	52.90	55.88
D	52.92	58.73	52.90	55.88
R	49.76	77.82	87.69	85.60
G	64.15	81.80	88.95	89.29

(c) Hits@5

	v1	v2	v3	v4
N	52.08	58.06	52.46	54.81
D	51.46	57.93	52.63	54.88
R	49.51	76.78	83.12	82.27
G	58.54	75.21	82.36	82.62

(d) Hits@1

	v1	v2	v3	v4
N	40.21	45.68	44.09	44.12
D	42.71	47.49	45.84	45.53
R	41.46	62.13	65.95	67.21
G	40.00	52.20	60.25	60.99

(e) MRR

	v1	v2	v3	v4
N	46.13	51.85	48.70	49.54
D	47.55	52.78	49.64	50.43
R	45.97	69.08	73.68	74.19
G	48.56	62.54	70.35	70.60

datasets (except some of the FB15k-237 datasets) in all metrics. At a closer inspection, the previous differentiable methods (Neural-LP and DRUM) perform significantly worse than GraIL. Moreover, as can be seen in Figure 5.1d, the strong structural inductive bias of GraIL enables it to be extremely parameter efficient with orders of magnitude less number of parameters. In comparison to the state-of-the-art statistical rule mining method, RuleN [29], GraIL does significantly better in all metrics on WN18RR and NELL-995 datasets. On FB15k-237 datasets, while it does better in AUC and Hits@10, RuleN takes over on the harder ranking metrics like Hits@1. An alternate view of this trend can be clearly noted in Figures 5.1b. This reinforces the fact that statistical rule-based methods are inherently very strong in Hits@1 and leaves room for improvements in our current approach.

Table 5.5: Inductive results on datasets derived from NELL-995. N denotes Neural-LP, D denotes DRUM, R denotes RuleN, G denotes GraIL

(a) AUC-PR

	v1	v2	v3	v4
N	64.66	83.61	87.58	85.69
D	59.86	83.99	87.71	85.94
R	84.99	88.40	87.20	80.52
G	**86.05**	**92.62**	**93.34**	**87.50**

(b) Hits@10

	WN18RR			
	v1	v2	v3	v4
N	52.92	58.94	52.90	55.88
D	52.92	58.73	52.90	55.88
R	49.76	77.82	87.69	85.60
G	64.15	81.80	88.95	89.29

(c) Hits@5

	v1	v2	v3	v4
N	52.08	58.06	52.46	54.81
D	51.46	57.93	52.63	54.88
R	49.51	76.78	83.12	82.27
G	58.54	75.21	82.36	82.62

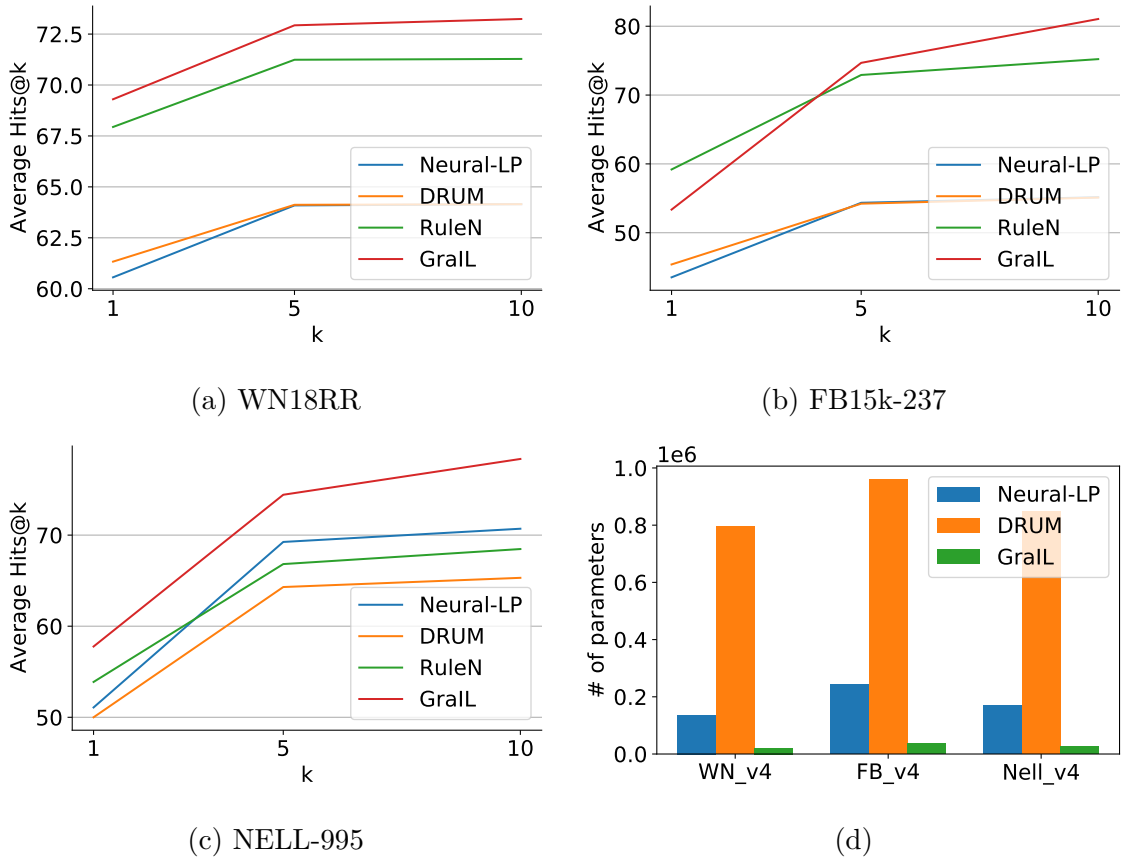
(d) Hits@1

	v1	v2	v3	v4
N	40.21	45.68	44.09	44.12
D	42.71	47.49	45.84	45.53
R	41.46	62.13	65.95	67.21
G	40.00	52.20	60.25	60.99

(e) MRR

	v1	v2	v3	v4
N	46.13	51.85	48.70	49.54
D	47.55	52.78	49.64	50.43
R	45.97	69.08	73.68	74.19
G	48.56	62.54	70.35	70.60

In aggregate, these results showcase the modelling superiority of the proposed model, GraIL, in comparison to existing differentiable methods and statistical rule-induction methods. This also indicates that GraIL is not only able to learn path-based logical rules, which are also learned by RuleN, but that GraIL is able to also exploit more complex structural patterns from the complete subgraph structure instead of just paths between the target nodes.



(a) WN18RR

(b) FB15k-237

(c) NELL-995

(d)

Figure 5.1: Figures 5.1a, 5.1b, 5.1c show the average Hits@k across all versions of the respective datasets. Figure 5.1d shows the number of parameters of all differentiable methods on version v4 of all datasets.

5.2 Transductive Relation Prediction

Transductive setting refers to the traditional setting of static graphs where there exists a fixed set of entities all of which are observed in training triplets. The models make predictions of new relations among these fixed set of entities (Figure 1.1b). Current state-of-the-art transductive models (Section 2.1) ground themselves to entity-specific embeddings, which gives them high degree of freedom to encode relational information. These models are trained so that entities 'closer' to each other in the knowledge graph have similar embeddings enabling them to exploit the positional information and homophily present in these graphs. In contrast, GraIL relies entirely on entity-independent structural information reflecting the multi-relational semantics.

As demonstrated, this results in a strong inductive bias to encode the logical rules and complex structural patterns underlying the knowledge graph. This, we believe, is complementary to the current state-of-the-art transductive embedding-based approaches. Based on this observation, in this set of experiments we explore (i) how GraIL performs in the transductive setting and (ii) the utility of ensembling GraIL with existing embedding-based approaches.

5.2.1 Datasets

The standard knowledge graph, as available in the literature, are setup in the transductive setting which we use as is. For NELL-995, we split the entire set of triplets into train/valid/test set by the ratio 70/15/15, making sure all the entities and relations in the valid and test splits occur at least once in the train set.

5.2.2 Models

As a representative set for embeddings-based methods, we use the models we described in Section 2.1 – TransE [15], DistMult [52], ComplEx [16], and finally the current state-of-the-art model RotatE [18]. For all the methods we use the implementation and hyperparameters provided by Sun et al.[18] [8] which gives state-of-the-art results on all methods. For a fair comparison of all the methods, we disable the self-adversarial negative sampling proposed by Sun et al.[18]. For GraIL, we use 2-hop neighborhood subgraphs for WN18RR and NELL-995, and 1-hop neighborhood subgraphs for FB15k-237. All the other hyperparameters remain the same as in the inductive setting.

[8] https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding

Table 5.6: Late fusion ensemble results on WN18RR (AUC-PR)

	TransE	DistMult	ComplEx	RotatE	GraIL
TransE	93.73	93.12	92.45	93.70	94.30
DistMult		93.08	93.12	93.16	95.04
ComplEx			92.45	92.46	94.78
RuleN				93.55	94.28
GraIL					90.91

Table 5.7: Late fusion ensemble results on NELL-995 (AUC-PR)

	TransE	DistMult	ComplEx	RotatE	GraIL
TransE	98.73	98.77	98.83	98.71	98.87
DistMult		97.73	97.86	98.60	98.79
ComplEx			97.66	98.66	98.85
RuleN				98.54	98.75
GraIL					97.79

5.2.3 Experimental Results

We mainly explore two different ensembling strategies – 1) late fusion, and 2) early fusion.

Late fusion. Our primary ensembling strategy is late fusion, i.e., ensembling the output scores of the constituent methods. We score each triplet with the methods that are to be ensembled. The scores output by each method form the feature vector for each data point. This feature vector is used to classify true triplets from false ones using a linear classifier. Intuitively, this linear classifier blends the predictions from all constitutes by learning a weighted aggregation of all the predictions. This 'blending' linear classifier is trained using the validation set of the respective datasets. At test time, the test set is scored by the constituent methods generating the feature vector which is then input to the linear classifier for the final score.

Tables 5.6, 5.7, and 5.8 show the AUC-PR performance of pairwise ensembling of different knowledge graph embedding (KGE) methods among themselves and with GraIL. A specific entry in these tables corresponds to the ensemble of pair of methods denoted by the row and column labels, with the individual performance of each

Table 5.8: Late fusion ensemble results on FB15k-237 (AUC-PR)

	TransE	DistMult	ComplEx	RotatE	GraIL
TransE	98.54	98.41	98.45	98.55	97.95
DistMult		97.63	97.87	98.40	97.45
ComplEx			97.99	98.43	97.72
RuleN				98.53	98.04
GraIL					92.06

Table 5.9: Late fusion ensemble results on WN18RR (Hits@10)

	TransE	DistMult	ComplEx	RotatE	GraIL
TransE	88.74	85.31	83.84	88.61	89.71
DistMult		85.35	86.07	85.64	87.70
ComplEx			83.98	84.30	86.73
RuleN				88.85	89.84
GraIL					73.12

Table 5.10: Late fusion ensemble results on NELL-995 (Hits@10)

	TransE	DistMult	ComplEx	RotatE	GraIL
TransE	98.50	98.32	98.43	98.54	98.45
DistMult		95.68	95.92	97.77	97.79
ComplEx			95.43	97.88	97.86
RuleN				98.09	98.24
GraIL					94.54

method on the diagonal. As can be seen from the last column of these tables, ensembling with GraIL resulted in consistent performance gains across all transductive methods in two out of the three datasets. Moreover, ensembling with GraIL resulted in more gains than ensembling any other two methods. Precisely, we define the gain obtained by ensembling two methods, $G(M_1, M_2)$, as follows

$$G(M_1, M_2) = \frac{P(M_1, M_2) - max(P(M_1), P(M_2))}{max(P(M_1), P(M_2)}.$$

In other words, it is the percentage improvement achieved relative to the best of the two methods. Thus, the average gain obtained by ensembling with GraIL is given

Table 5.11: Late fusion ensemble results on FB15k-237 (Hits@10)

	TransE	DistMult	ComplEx	RotatE	GraIL
TransE	98.87	98.96	99.05	98.87	98.71
DistMult		98.67	98.84	98.86	98.41
ComplEx			98.88	98.94	98.64
RuleN				98.81	98.66
GraIL					75.87

Table 5.12: Relative gain of pairwise ensembling AUC-PR

	WN18RR		FB15k-237		NELL-995	
	AUC-PR	Hits@10	AUC-PR	Hits@10	AUC-PR	Hits@10
$G_{\text{avg}}^{\text{GraIL}}$	1.5%	2.06%	0%	0%	0.62%	1.23%
$G_{\text{avg}}^{\text{KGE}}$	0.007%	0.14%	0.002%	0.06%	0.08%	0.05%

by

$$G_{\text{avg}}^{\text{GraIL}} = \frac{1}{4} \sum_{|M_1| \in KGE} G(M_1, \text{GraIL}),$$

and the average gain obtained by pairwise ensembling among the KGE embedding methods is given by

$$G_{\text{avg}}^{\text{KGE}} = \frac{1}{12} \sum_{(|M_1|,|M_2|) \in KGE} G(M_1, M_2).$$

The relative gains on all datasets are summarized in Table 5.12. The average gains in AUC-PR obtained by GraIL ($G_{\text{avg}}^{\text{GraIL}}$) on WN18RR and NELL-995 are 1.5% and 0.62%, respectively. This is orders of magnitude better than the average gains obtained by KGE ensembling ($G_{\text{avg}}^{\text{KGE}}$): 0.007% and 0.08%. Surprisingly, none of the ensemblings resulted in significant gains on FB15k-237. Thus, while GraIL on its own is optimized for the inductive setting and not state-of-the-art for transductive prediction, it does give a meaningful improvement over state-of-the-art transductive methods via ensembling. Tables 5.9, 5.10, and 5.11 show similar trends in improvements obtained in terms of Hits@10.

[8]The Hits@10 numbers in these tables are higher than usually reported in the literature due to the approximation we adopt.

Early fusion. Early fusion refers to using the embeddings learnt by transductive methods as additional input into GraIL. In particular, the node labels, as computed by our original node-labeling scheme, are concatenated with node-embeddings learnt by a transductive method (TransE, in this case). The addition of these pre-trained embeddings results in as much as 21.1% performance boost in Hits@10 performance (Table 5.13).

Table 5.13: Early fusion ensemble with TransE results

	WN18RR		FB15k-237		NELL-995	
	AUC-PR	Hits@10	AUC-PR	Hits@10	AUC-PR	Hits@10
GraIL	90.91	73.12	92.06	75.87	97.79	94.54
GraIL++	96.20	88.59	93.91	89.68	98.11	97.93

Thus, while *late fusion* demonstrates the complementary inductive bias that GraIL embodies, this kind of *early fusion* demonstrates the natural ability of GraIL to leverage any node embeddings/features available.

5.3 Additional analysis

In this section we present ablation studies and hyperparameter sensitivity of GraIL. These experiments provide empirical evidence for some of the theoretical motivations that inspired the design parameters of our method.

5.3.1 Ablation Study

We conduct ablation experiments to emphasize the importance of the three key components of GraIL: i) enclosing subgraph extraction ii) double radius node labeling scheme, and iii) attention in the GNN. The results are summarized in Table 5.14.

Enclosing subgraph extraction. As mentioned in Section 4.1.1, we assume that the logical evidence for a particular link can be found in the subgraph surrounding the two target nodes of the link. Thus we proposed to extract the subgraph induced

by all the nodes occurring on a path between the two target nodes. Here, we want to emphasize the importance of extracting only the paths as opposed to a more naive choice of extracting the subgraph induced by all the k-hop neighbors of the target nodes. The performance drastically drops in such a configuration (second row of Table 5.14). While we expected our model to leverage the entire subgraph to learn about the target entities and the relation, we observe that across all the datasets the model catastrophically overfits to the training data when presented with full subgraph.

Double radius node labeling. In Section 4.1.2, we introduced a node-labelling scheme that captures the topological position of each node with respect to the target nodes and reflects its structural role in the subgraph. In fact, the proof of Theorem 1 assumes having uniquely labeled target nodes, u and v, which was a property of the proposed node-labelling scheme. We highlight the importance of this by evaluating GraIL with constant node labels of $(1, 1)$ instead of the originally proposed scheme. As can be noted from third row of Table 5.14, the drop in performance emphasizes the importance of our node-labeling scheme in helping GraIL find the logical paths.

Attention in the GNN. Inspired by recently introduced attention mechanisms in aggregating information from neighboring nodes, we designed an attention mechanism as a function of the target entities, target relation and the subgraph around the target entities. As noted in the proof of Theorem 1, this attention mechanism is a vital component of our model in encoding the path rules. In particular, it allows the model to indicate whether a particular relation is incident to a particular entity and enabling the model to detect thee existence of a particular path between the target nodes. We evaluate GraIL without this attention mechanism and note significant performance drop (fourth row in Table 5.14), which echos with our theoretical findings.

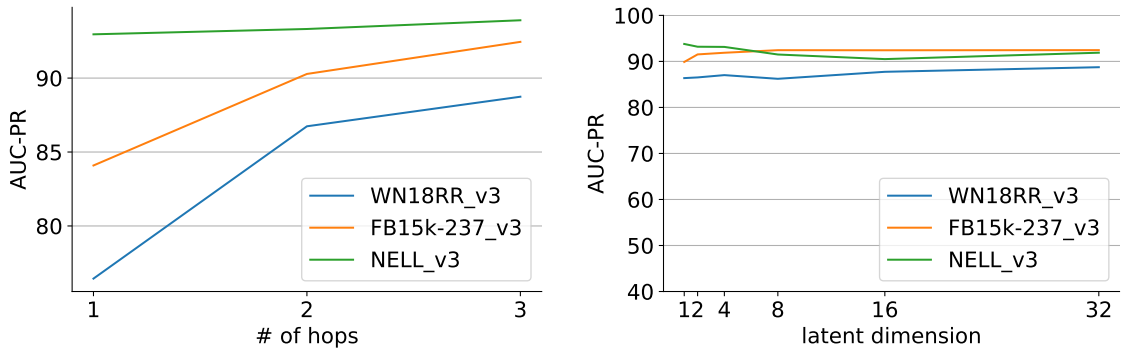
Table 5.14: Ablation study of the proposed framework (AUC-PR)

	FB (v3)	NELL (v3)
GraIL	**91.68**	**93.34**
GraIL w/o enclosing subgraph	84.25	85.89
GraIL w/o node labeling scheme	82.07	84.46
GraIL w/o attention in GNN	90.27	87.30

5.3.2 Hyperparameter sensitivity analysis

In this section we highlight the sensitivity of GraIL with respect to different hyper-parameters and substantiate some of our intuitions developed in our the theoretical findings and the experiments so far.

Size of enclosing subgraph. Observation 1 states that given the enclosing subgraph from the k-hop neighborhood GraIL learns path-rules of length $k+1$. Naturally, one would expect the performance to improve as we allow the model to learn longer rules and hence capture more complexity. Figure 5.2a shows how the performance of GraIL increases as we increase the neighborhood size. Note that the size of the subgraph can increase exponentially as we increase the number of hops. As described in Section 4.3, this heavily impacts the run-time of our model and inhibits us from increasing the number of hops at free will.



(a) (b)

Figure 5.2: Sensitivity to (a) neighborhood size of enclosing subgraphs, and (b) latent dimension of GraIL

Latent dimension of GraIL. One peculiar property of GraIL, as noted in Theorem

1, is its provable representational capacity with just 1-dimensional latent features.[9] Figure 5.2b shows how the performance depends on the latent dimension. We note that performance is relatively stable with marginal improvement or deterioration with respect to the latent dimension. This conforms with our theoretical analysis that a dimension of 1 is sufficient to encode path-based logical rules that models like RuleN induce.

[9]Note that, for simplicity, we tie all the latent dimensions of GraIL to be the same value.

6

Conclusion

Graph neural networks (GNN) present a new class of methods, which have recently proved effective in many relational learning tasks. In this work, we theoretically prove that GNNs have the representational capacity to encode a useful subset of first order logic formulae (chain-like Horn clauses). Based on this, we proposed a GNN-based framework, GraIL, for knowledge graph reasoning where the underlying relational semantics can be effectively captured as Horn clauses. Unlike embedding-based approaches, GraIL captures entity-independent relational semantics and hence can naturally generalize to entirely new graphs with nodes that were unseen during training. Moreover, we showed that GraIL brings an inductive bias complementary to the current state-of-the-art knowledge graph completion methods. In particular, we demonstrated, with a thorough set of experiments, performance boosts to various knowledge graph embedding methods when ensembled with GraIL.

Although the relational learning abilities are showcased in the context of knowledge graphs, the theoretical findings in this work have implications that reach beyond just knowledge graphs. This work brings forth a new fundamental expressive power of GNNs, i.e., the ability to learn logical rules, that can be applied to relational reasoning in domains like scene understanding, question answering, commonsense reasoning, etc.

6.1 Limitations

Our approach, GraIL, as presented in this work has two key limitations in comparison to existing methods: 1) computational complexity, and 2) intrepretability.

6.1.1 Computational complexity

As pointed out in Section 4.3, the runtime of GraIL, in practice, in dominated by the node-labelling scheme that computes the shortest distance of every node in the enclosing subgraph from the target nodes (via Dijkstra's algorithm). This limitation inhibits us from adopting the standard evaluation setup of ranking all entities and forces us to approximate these metrics via sampling schemes. Although, in practice, smaller and more relevant relation-specific candidate sets can be derived using type constraints and meta-data available [66], these subsampled metrics cannot be easily compared with the full ranking metrics in the literature. In this regard, we note that GraIL can be adapted to efficiently perform full ranking, e.g., by using an ensemble strategy where-in GraIL re-scores a faster but weaker models' (for e.g., TransE) top-100 predictions. As a proof-of-concept: on WN18RR, we were able to improve the overall MRR (with full ranking) from 20.26% (TransE baseline) to 33.52% using this approach.

6.1.2 Interpretability

Although GraIL outperforms existing inductive approaches, one of the key disadvantages of our methods over other approaches is its lack of interpretability. In particular, all the inductive baselines—Neural-LP, DRUM, and RuleN—derive readable logical rules that makes them makes them more transparent and reliable. This lack of interpretability also limits our understanding of the kind of rules and structures GraIL is able to learn over methods like RuleN (which in theory is just as powerful). In that

regard, an interesting direction to investigate is leveraging recent works on explaining predictions made by GNNs [113] to get a better understanding of GraIL.

6.2 Future directions

With a comprehensive study of existing methods for inductive relation prediction and a set of new benchmark datasets we open up a new direction for exploration on inductive reasoning in the context of knowledge graphs. Moreover, our findings of connection between logical reasoning and graph neural network complement and/or augment many recent works and point to interesting avenues of research.

6.2.1 Frontiers of GNNs and logical reasoning

Many recent works have concurrently explored the connections of logical reasoning and graph neural networks. Barceló et al.[27] presented a strong connection between the expressive powers of GNNs and a subset of first order predicate logic—FOC_2—leveraging their connections to WL isomorphism test [114]. However, their results are centered around simple graphs and unary logical formulas. Our results augment their findings by demonstrating similar connections in the context of multi-relational graphs and binary logical formulas. One way to interpret our results in their formulation is to use the labels of the nodes as unary predicates and relations as binary predicates to represent the Horn clauses within FOC_2 formulation. This interpretation, like our proof of Theorem 1, relies on our unique labelling scheme. It would be interesting to generalize this ability by simplifying the node-labelling scheme. Such a simplification would also alleviate the computational complexity of GraIL and make it more scalable.

Another line of concurrent work combines graph neural networks with powerful probabilistic deduction engines–Markov Logic Networks [78, 79]. These approaches focus on the performing deduction and inference on knowledge graphs using a given set of pre-defined rules. In fact, Y. Zhang et al.[79] uses Neural-LP (one of our

inductive baselines) in their pre-processing step to derive logical rules. Our approach is complimentary to theirs in that we implicitly perform rule induction along with deduction and inference. However, these works offer an advantage of having the ability to incorporate domain-expert rules in their inference engine. A natural direction for future work is to consider ways to combine these two approaches and design end-to-end rule learning induction and inference engines. This also highlights the need to extract rules from GraIL one if its current limitations as described earlier.

6.2.2 Inductive relation prediction

We propose the first comprehensive benchmark datasets for inductive relation prediction in knowledge graphs. One of the interesting properties of these datasets is that there can be a shift in distribution of relations when going from training graph to the test graph. With such a shift, it can be challenging to transfer knowledge on relations that have a very low frequency in the training graph. This is the same challenge that motivates many few-shot link prediction works [66, 85, 111]. One can leverage meta-learning strategies [111] to help mitigate this challenge in the inductive setting and perhaps combine the structural inductive bias GraIL to help improve learning in the few-shot relational learning setting [66, 85].

Bibliography

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *Proceedings of International Conference on Learning Representations*. 2014.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *Proceedings of the IEEE Signal Processing Magazine* (2012).

[4] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[5] Drew A Hudson and Christopher D Manning. "Gqa: A new dataset for real-world visual reasoning and compositional question answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[6] Antoine Bordes, Jason Weston, and Nicolas Usunier. "Open question answering with weakly supervised embedding models". In: *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2014.

[7] Antoine Bordes, Sumit Chopra, and Jason Weston. "Question Answering with Subgraph Embeddings". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2014.

[8] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. "Semantic parsing on freebase from question-answer pairs". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2013.

[9] Larry Heck, Dilek Hakkani-Tür, and Gokhan Tur. "Leveraging Knowledge Graphs for Web-Scale Unsupervised Semantic Parsing". In: *Proceedings of Interspeech*. 2013.

[10] Danica Damljanovic and Kalina Bontcheva. "Named entity disambiguation using linked data". In: *Proceedings of the Extended Semantic Web Conference*. 2012.

[11] Zhicheng Zheng, Xiance Si, Fangtao Li, Edward Y Chang, and Xiaoyan Zhu. "Entity disambiguation with freebase". In: *Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*. 2012.

[12] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. "Knowledge-based weak supervision for information extraction of overlapping relations". In: *Proceedings of the Association for Computational Linguistics: Human Language Technologies*. 2011.

[13] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. "Improving efficiency and accuracy in multilingual entity extraction". In: *Proceedings of the International Conference on Semantic Systems*. 2013.

[14] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. "A Review of Relational Machine Learning for Knowledge Graphs". In: *Proceedings of the IEEE* (2016).

[15] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. "Translating Embeddings for Modeling Multi-relational Data". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2013.

[16] Théo Trouillon, Christopher R. Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. "Knowledge Graph Completion via Complex Tensor Factorization". In: *Proceedings of the Journal of Machine Learning Research* (2017).

[17] Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. "Convolutional 2D Knowledge Graph Embeddings". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018.

[18] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. "RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space". In: *Proceedings of the International Conference on Learning Representations*. 2019.

[19] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. "Revisiting Semi-Supervised Learning with Graph Embed dings". In: *Proceedings of the International Conference on Machine Learning*. 2016.

[20] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *Proceedings of International Conference on Learning Representations*. 2017.

[21] William L. Hamilton, Rex Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2017.

[22] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. "Learning a SAT Solver from Single-Bit Supervision". In: *Proceedings of the International Conference on Learning Representations*. 2019.

[23] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[24] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. "Neural Relational Inference for Interacting Systems". In: *Proceedings of International Conference on Machine Learning*. 2018.

[25] Ferran Alet, Adarsh K. Jeewajee, Maria Bauza, Alberto Rodríguez, Tomas Lozano-Perez, and Leslie Pack Kaelbling. "Graph Element Networks: adaptive, structured computation and memory". In: *Proceedings of International Conference on Machine Learning*. 2019.

[26] Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. "Neural Execution of Graph Algorithms". In: *Proceedings of the International Conference on Learning Representations*. 2020.

[27] Pablo Barceló, Egor V. Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan Pablo Silva. "The Logical Expressiveness of Graph Neural Networks". In: *Proceedings of the International Conference on Learning Representations*. 2020.

[28] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. "AMIE: Association Rule Mining Under Incomplete Evidence in Ontological Knowledge Bases". In: *Proceedings of the International World Wide Web Conference*. 2013.

[29] Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. "Fine-Grained Evaluation of Rule- and Embedding-Based Systems for Knowledge Graph Completion". In: *Proceedings of the International Semantic Web Conference*. 2018.

[30] Yuan Yang and Le Song. "Learn to Explain Efficiently via Neural Logic Inductive Learning". In: *Proceedings of the International Conference on Learning Representations*. 2020.

[31] Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. "Differentiable Reasoning over a Virtual Knowledge Base". In: *Proceedings of the International Conference on Learning Representations*. 2020.

[32] Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. "Neural Module Networks for Reasoning over Text". In: *Proceedings of the International Conference on Learning Representations*. 2020.

[33] Fan Yang, Zhilin Yang, and William W Cohen. "Differentiable learning of logical rules for knowledge base reasoning". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2017.

[34] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. "DRUM: End-To-End Differentiable Rule Mining On Knowledge Graphs". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2019.

[35] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. "Revisiting Semi-Supervised Learning with Graph Embeddings". In: *Proceedings of the International Conference on Machine Learning*. 2016.

[36] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. "Neural message passing for quantum chemistry". In: *Proceedings of the International Conference on Machine Learning*. 2017.

[37] William L. Hamilton, Rex Ying, and Jure Leskovec. “Representation Learning on Graphs: Methods and Applications”. In: *Proceedings of the IEEE Data Engineering Bulletin* (2017).

[38] Kristina Toutanova and Danqi Chen. “Observed versus latent features for knowledge base and text inference”. In: *Workshop on Continuous Vector Space Models and their Compositionality*. 2015.

[39] Maximilian Nickel, Xueyan Jiang, and Volker Tresp. “Reducing the Rank in Relational Factorization Models by Including Observable Patterns”. In: *Proceedings of the Advances in Neural Information Processing Systems 27*. 2014.

[40] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. “The graph neural network model”. In: *IEEE Transactions on Neural Networks* (2008).

[41] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. “Geometric deep learning: going beyond euclidean data”. In: *IEEE Signal Processing Magazine* (2017).

[42] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. “A semantic matching energy function for learning with multi-relational data”. In: *Machine Learning* (2014).

[43] Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka, Partha P. Talukdar, Bo Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matt Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. “Never-Ending Learning”. In: *Communications of the ACM* (2015).

[44] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. "Graph convolutional neural networks for web-scale recommender systems". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018.

[45] Linhong Zhu, Majid Ghasemi-Gol, Pedro Szekely, Aram Galstyan, and Craig A Knoblock. "Unsupervised entity resolution on multi-type graphs". In: *Proceedings of the International Semantic Web Conference*. 2016.

[46] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. "Linguistic Regularities in Continuous Space Word Representations". In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2013.

[47] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. *Knowledge Graph Embedding by Translating on Hyperplanes*. 2014.

[48] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. "Learning Entity and Relation Embeddings for Knowledge Graph Completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2015.

[49] Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. "STransE: a novel embedding model of entities and relationships in knowledge bases". In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016.

[50] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. "Knowledge Graph Embedding via Dynamic Mapping Matrix". In: *Proceedings of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*. 2015.

[51] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. "A three-way model for collective learning on multi-relational data." In: *Proceedings of the International Conference on Machine Learning*. 2011.

[52] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. "Embedding Entities and Relations for Learning and Inference in Knowledge Bases". In: *Proceedings of the International Conference on Learning Representations*. 2015.

[53] Seyed Mehran Kazemi and David Poole. "Simple embedding for link prediction in knowledge graphs". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2018.

[54] Ivana Balazevic, Carl Allen, and Timothy Hospedales. "TuckER: Tensor Factorization for Knowledge Graph Completion". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*. 2019.

[55] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. "Reasoning with neural tensor networks for knowledge base completion". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2013.

[56] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. "Holographic embeddings of knowledge graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2016.

[57] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. "Modeling Relational Data with Graph Convolutional Networks". In: *Proceedings of the Extended Semantic Web Conference*. 2017.

[58] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. "Composition-based Multi-Relational Graph Convolutional Networks". In: *Proceedings of the International Conference on Learning Representations*. 2020.

[59] Diego Marcheggiani and Ivan Titov. "Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2017.

[60] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. "End-to-End Structure-Aware Convolutional Networks for Knowledge Base Completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2019).

[61] Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. "A Vectorized Relational Graph Convolutional Network for Multi-Relational Network Alignment". In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 2019.

[62] Léon Bottou, Frank E Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning". In: *Society for Industrial and Applied Mathematics Review* (2018).

[63] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2013.

[64] Bhushan Kotnis and Vivi Nastase. "Analysis of the impact of negative sampling on link prediction in knowledge graphs". In: *arXiv preprint arXiv:1708.06816* (2017).

[65] Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. "Embedding logical queries on knowledge graphs". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2018.

[66] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. "One-Shot Relational Learning for Knowledge Graphs". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2018.

[67] Luc Dehaspe and Hannu Toivonen. "Discovery of frequent datalog patterns". In: *Data Mining and Knowledge Discovery* (1999).

[68] Stephen Muggleton. “Inverse entailment and Progol”. In: *New Generation Computing* (1995).

[69] Quan Wang, Bin Wang, and Li Guo. “Knowledge base completion using embeddings and rules”. In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 2015.

[70] Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. “Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances”. In: *Proceedings of the Conference on Information and Knowledge Management*. 2015.

[71] Matthew Richardson and Pedro Domingos. “Markov logic networks”. In: *Machine Learning* (2006).

[72] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. “Jointly embedding knowledge graphs and logical rules”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2016.

[73] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. “Knowledge graph embedding with iterative guidance from soft rules”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018.

[74] Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. “Lifted Rule Injection for Relation Embeddings”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2016.

[75] Pasquale Minervini, Luca Costabello, Emir Muñoz, Vıt Nováček, and Pierre-Yves Vandenbussche. “Regularizing knowledge graph embeddings via equivalence and inversion axioms”. In: *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2017.

[76] Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. "Adversarial Sets for Regularising Neural Link Predictors". In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (2017).

[77] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. "Injecting logical background knowledge into embeddings for relation extraction". In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015.

[78] Meng Qu and Jian Tang. "Probabilistic logic neural networks for reasoning". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2019.

[79] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. "Efficient Probabilistic Logic Reasoning with Graph Neural Networks". In: *Proceedings of International Conference on Learning Representations*. 2020.

[80] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. "Representation learning of knowledge graphs with entity descriptions". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2016.

[81] Yu Zhao, Sheng Gao, Patrick Gallinari, and Jun Guo. "Zero-Shot Embedding for Unseen Entities in Knowledge Graph". In: *The Institute of Electronics, Information and Communication Engineers Transactions on Information and Systems* (2017).

[82] Baoxu Shi and Tim Weninger. "Open-world knowledge graph completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018.

[83] Peifeng Wang, Jialong Han, Chenliang Li, and Rong Pan. "Logic Attention Based Neighborhood Aggregation for Inductive Knowledge Graph Embedding". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019.

[84] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. "Knowledge Transfer for Out-of-knowledge-base Entities: A Graph Neural Network Approach". In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 2017.

[85] Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. "Meta Relational Learning for Few-Shot Link Prediction in Knowledge Graphs". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*. 2019.

[86] Xin Lv, Yuxian Gu, Xu Han, Lei Hou, Juanzi Li, and Zhiyuan Liu. "Adapting Meta Knowledge Graph Information for Multi-Hop Reasoning over Few-Shot Relations". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*. 2019.

[87] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *Proceedings of the International Conference on Machine Learning*. 2017.

[88] Julien Leblay and Melisachew Wudage Chekol. "Deriving validity time in knowledge graph". In: *Companion Proceedings of the The Web Conference 2018*. 2018.

[89] Sam De Winter, Tim Decuypere, Sandra Mitrović, Bart Baesens, and Jochen De Weerdt. "Combining temporal aspects of dynamic networks with Node2Vec for a more efficient dynamic link prediction". In: *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2018.

[90] Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. "TEQUILA: Temporal question answering over knowledge

bases". In: *Proceedings of the ACM International Conference on Information and Knowledge Management*. 2018.

[91] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs". In: *Proceedings of the International Conference on Machine Learning*. 2017.

[92] Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. "Diachronic Embedding for Temporal Knowledge Graph Completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.

[93] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. "Variational graph recurrent neural networks". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2019.

[94] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation Learning on Graphs with Jumping Knowledge Networks". In: *Proceedings of the International Conference on Machine Learning*. 2018.

[95] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How Powerful are Graph Neural Networks?" In: *Proceedings of the International Conference on Learning Representations*. 2019.

[96] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks". In: *Proceedings of the International Conference on Learning Representations* (2018).

[97] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks". In: *arXiv preprint arXiv :1505.00387*. 2015.

[98] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep residual learning for image recognition. CoRR abs/1512.03385 (2015)*. 2015.

[99] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. "Column networks for collective classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2017.

[100] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. "Gated graph sequence neural networks". In: *Proceedings of the International Conference on Learning Representations*. 2016.

[101] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2014.

[102] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* (1997).

[103] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. "An end-to-end deep learning architecture for graph classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018.

[104] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. "Hierarchical graph representation learning with differentiable pooling". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2018.

[105] Cătălina Cangea, Petar Veličković, Nikola Jovanović, Thomas Kipf, and Pietro Liò. "Towards sparse hierarchical graph classifiers". In: *arXiv preprint arXiv :1811.01287* (2018).

[106] Balasubramaniam Srinivasan and Bruno Ribeiro. "On the Equivalence between Positional Node Embeddings and Structural Graph Representations". In: *Proceedings of the International Conference on Learning Representations*. 2020.

[107] Muhan Zhang and Yixin Chen. "Link prediction based on graph neural networks". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2018.

[108] Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. "CLUTRR: A diagnostic benchmark for inductive reasoning from text". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2019.

[109] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* (1991).

[110] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. "Representing text for joint embedding of text and knowledge bases". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2015.

[111] Wenhan Xiong, Thien Hoang, and William Yang Wang. "DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2017.

[112] William W. Cohen. "TensorLog: A Differentiable Deductive Database". In: *arXiv preprint arXiv :1605.06523* (2016).

[113] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. "Gnnexplainer: Generating explanations for graph neural networks". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2019.

[114] Jin-Yi Cai, Martin Fürer, and Neil Immerman. "An optimal lower bound on the number of variables for graph identification". In: *Combinatorica* (1992).