Multi-Hop Relation Aware Representations for Inductive Knowledge Graphs

Aniruddha Bala[*] **Ankit Sharma**[*] **Shlok Sharma** **Pinaki Bhaskar**
{aniruddha.b, ank.sharma, shlok.sharma, pinaki.b}@samsung.com
Advanced Technology Lab
Samsung Research Institute, Bangalore

Abstract

Recent knowledge graph (KG) embedding methods explore parameter-efficient representations for large-scale KGs. These techniques learn entity representation using a fixed size vocabulary. Such a vocabulary consists of all the relations and a small subset of the full entity set, referred to as anchors. An entity is hence expressed as a function of reachable anchors and immediate relations. The performance of these methods is, therefore, largely dependent on the entity tokenization strategy. Especially in inductive settings, the representation capacity of these embeddings is limited due to the absence of anchor entities, as unseen entities have no connection with training graph entities.

In this work, we propose a novel entity tokenization strategy that tokenizes an entity into a set of anchors based on relation similarity and relational paths. Our model MH-RARe overcomes the challenge of unseen entities not being directly connected to the anchors by selecting informative anchors from the training graph using relation similarity. Experiment results show that our model outperforms the baselines on multiple datasets for inductive knowledge graph completion task, attaining upto 5% improvement, while maintaining parameter efficiency.

1 Introduction

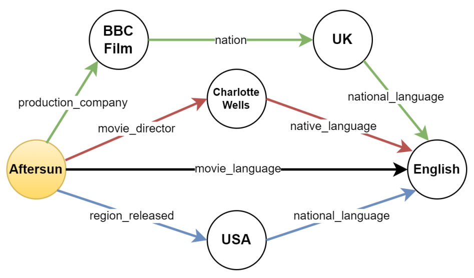
Knowledge Graphs (KGs) are powerful structures that consist of a vast collection of interconnected facts, presented in the form of a graph. Each fact is represented as a triplet in the form $\langle \texttt{subject}, \texttt{relation}, \texttt{object} \rangle$. To effectively utilize KGs, representation learning plays a crucial role. In this context, representation learning involves learning a d-dimensional vector representation for every entity and relation in KG. Typically, these representations are learned through a link prediction task, which aims to predict the missing links in the KG.

Traditional KG embedding methods employ shallow embedding techniques, wherein entities and relations are mapped onto a lower-dimensional vector space. These methods rely on an embedding lookup function that ensures each entity and relation is uniquely associated with a d-dimensional vector. However, these methods encounter challenges when dealing with large-scale knowledge graphs, as the number of parameters required scales linearly with the size of the entity set. Also, these methods are applicable only to transductive settings, where the set of entities remains fixed. Nevertheless, real world KGs evolve as new entities get added to the graph over time. Link prediction on these graphs requires the ability to generalize to these unseen entities. Inductive link prediction is the task of inferring missing links in such graphs.

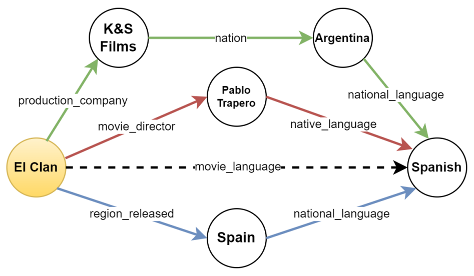
Recently, motivated by the compositional models to represent words such as WordPiece (Schuster and Nakajima, 2012), NodePiece (Galkin et al., 2022) proposes a framework to limit the vocabulary size in KGs. This is done by limiting the vocabulary to a fixed set of entities (called as anchors) and relations in the KG. An entity in the KG is then tokenized into it's nearest anchors and immediate relations. While anchors are available for seen entities, unseen entities are represented just using immediate relations as the entities in the anchor set are disconnected from these unseen entities. This limits the expressive ability of the unseen entities.

In this work we address this limitation and propose a novel anchor selection strategy for unseen entities in an inductive KG. As shown by previous work GAKE (Feng et al., 2016), immediate relations of an entity in a KG, can reveal the nature of an entity. We use this result to draw m relationally similar anchor nodes to a given unseen entity from the training graph. Our hypothesis is that, such re-

[*]Both authors contributed equally to this work.



(a) Train Graph



(b) Test Graph

Figure 1: **Inductive Inference Example.** Figure shows two disjoint train and test subgraphs for movies Aftersun and El Clan. To predict ⟨`El Clan`, `movie_language`, ?⟩ the entity `Aftersun` can serve as a meaningful anchor as both entities have similar subgraph structure.

lationally similar nodes may contain some valuable information that will aid in better link prediction. Furthermore, rather than looking at 1-hop immediate relations, we tokenize an entity based on n relation paths originating from that entity which helps in building a richer relational context for an entity. Our experiment results provide substantial evidence to support our hypothesis, indicating such a tokenization strategy can improve link prediction performance in inductive link prediction tasks.

2 Related Work

Conventional embedding based methods: Traditional approaches to embedding methods for link prediction in knowledge graphs have been widely explored in the literature. These techniques aim to capture the inherent semantic relationships and structural patterns in the graph data to predict missing links between entities. Approaches like TransE (Bordes et al., 2013), TransH (Wang et al., 2014, 2020) and TransR (Lin et al., 2015) assume that the embeddings of entities and relations follow some geometric rules in vector space. Other approaches use bilinear forms or complex-valued operations to model the interactions between entity and relation embeddings. Ex. Distmult (Yang et al., 2015) uses simple dot product between entities and relations to calculate plausibility of the triple. ComplEx (Trouillon et al., 2017) uses Hermitian dot product to calculate plausibility of a true triple as it represents relations as complex-valued vectors. These embedding methods provide effective means for link prediction in transductive setting but they can not generalize to unseen entities, so we cannot use them in inductive setting.

Inductive embeddings based methods: Some recent works have shown how to create embeddings for nodes that are not seen before, but they have some limitations. GraphSAGE (Hamilton et al., 2017) and Graph2Gauss (Bojchevski and Günnemann, 2018) depend on having node features that are often missing in many KGs. LAN (Wang et al., 2019) and (Hamaguchi et al., 2017) produce embeddings for new nodes by combining the embeddings of nearby nodes using GNNs. However, both of these methods require the new nodes to have known nodes around them and cannot deal with completely new graphs.

Rule-induction based methods: These methods are different from embedding-based methods because they infer probabilistic logical-rules from the knowledge graph by finding statistical patterns and regularities in it, like (Meilicke et al., 2018a) and (Galárraga et al., 2013). These methods can naturally handle unseen nodes because the rules do not depend on specific node identities, but they have problems with scalability and expressiveness due to their reliance on rules. Inspired by these statistical rule-induction methods, the NeuralLP model (Yang et al., 2017) uses TensorLog (Cohen, 2016) operators to learn logical rules from KGs in a way that is fully differentiable. Sadeghian et al. (2019) improved on NeuralLP and proposed DRUM, which learns more precise rules. We compare our methods with this group of methods in the inductive setting.

GNN based link prediction: Some recent methods have proposed to use Graph Neural Network based models for link prediction task. They first take out a subgraph around each target link based on the k-hop neighbourhood of the target entities, encode the subgraphs using a Graph Neural Network (GNN), then learn a function that maps subgraph structural patterns to link existence. For instance, R-GCN (Schlichtkrull et al., 2017) employed relational graph convolutional layers to collect information from neighboring nodes and edges. CompGCN (Vashishth et al., 2020) utilized composition functions to merge entity and relation embeddings prior to aggregation. However, these methods are applicable to only transductive settings.

Some recent methods such as GraIL (Teru et al., 2020), TACT (Chen et al., 2021), CoMPILE(Mai et al., 2020), ConGLR(Lin et al., 2022) utilize GNNs for inductive link prediction. GraIL first encodes the subgraph around a link using the double radius node labelling scheme and then applies a RGCN based GNN to predict the plausibility of the link. TACT extends GraIL by adding topological patterns while CoMPILE improves the message passing between entities through a communicative kernel. Along with the context graphs, ConGLR uses logical reasoning to improve inductive link prediction performance. Although these methods have achieved good performance on inductive link prediction, they do not fully consider the global context of the KG, which could provide valuable insights for link prediction.

Parameter efficient link prediction: Conventional KG embedding methods store embeddings for all entities and relations. However, NodePiece (Galkin et al., 2022) introduced a unique approach by leveraging word tokenization techniques and applying them to knowledge graphs for node tokenization.

In our work, we build upon the foundations of NodePiece and further enhance it for inductive link prediction tasks. While NodePiece does not utilize anchors in the inductive setting, we propose a novel approach that incorporates anchors for unseen entities by considering the outgoing relations from the entity.

3 Proposed Method

We denote a graph $\mathcal{G}$ as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations and $\mathcal{T}$ is the set of observed triples $\mathcal{T} = \{(h, r, t) \mid (h, r, t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$. Given a training graph $\mathcal{G}_{train} = (\mathcal{E}_{train}, \mathcal{R}_{train}, \mathcal{T}_{train})$, the task of inductive inference is to infer missing links in test graph $\mathcal{G}_{test} = (\mathcal{E}_{test}, \mathcal{R}_{test}, \mathcal{T}_{test})$, where, the train and test graphs are disjoint $\mathcal{E}_{train} \cap \mathcal{E}_{test} = \phi$, and relations in the test graph is a subset of relations in the training graph $\mathcal{R}_{test} \subseteq \mathcal{R}_{train}$. We augment train and test graphs with inverse relations to ensure connectivity from each node.

The idea of learning parameter efficient representations in knowledge graphs is to represent entities as a function of elements (entities and relations) from a smaller vocabulary set. Typically in KGs the number of entities far exceeds the number of relations. Therefore we form our vocabulary with a

Algorithm 1: Anchor selection algorithm

```
Input  : Training graph
         $\mathcal{G}_{train} = (\mathcal{E}_{train}, \mathcal{R}_{train}, \mathcal{T}_{train})$,
         number of anchors $m$
Output : Anchor Set $\mathcal{A}$
1  $e2r = dict()$
2  $r2e = dict()$
3  $e2r_{count} = heapdict()$
4  $\mathcal{A} = set()$
5  for $(h, r, t) \in \mathcal{T}_{train}$ do
6      $e2r[h].add(r)$
7      $r2e[r].add(h)$
8  end for
9  while $\mid \mathcal{A} \mid < m$ do
10     $e2r_{count}.clear()$
11     for $e \in e2r.keys()$ do
12         if $e \notin \mathcal{A}$ then
13             $e2r_{count}[e] = -len(e2r[e])$
14         end if
15     end for
16     $\mathcal{R}_{rem} = \mathcal{R}_{train}$
17     $r_{total} = \mid \mathcal{R}_{train} \mid$
18     while $r_{total} > 0$ and $len(e2r_{count}) > 0$
        do
19         if $\mid \mathcal{A} \mid == m$ then
20             return $\mathcal{A}$
21         end if
22         $e, r_{count} = e2r_{count}.popitem()$
23         if $r_{total} == 0$ or $r_{count} == 0$ then
24             break
25         end if
26         $\mathcal{A} = \mathcal{A} \cup \{e\}$
27         for $r \in e2r[e]$ do
28             if $r \in \mathcal{R}_{rem}$ then
29                 $\mathcal{R}_{rem} = \mathcal{R}_{rem} \setminus \{r\}$
30                 for $e \in r2e[r]$ do
31                     $e2r_{count}[e] =$
                         $e2r_{count}[e] + 1$
32                 end for
33             end if
34         end for
35         $r_{total} = r_{total} + r_{count}$
36     end while
37 end while
```

smaller subset of entities called as the anchor set $\mathcal{A}$. Our objective here is to learn an encoder function $f(\{a_i\}^m, \{p_j\}^n)$ for any entity $e \in \mathcal{E}_{train} \cup \mathcal{E}_{test}$, where $\{a_i\}^m$ is the set of m anchors picked for e and $\{p_j\}^n$ is the set of n paths originating from e. m and n are hyperparameters of our model their optimal values can be found in the Appendix. In the subsequent sections we elaborate on this process of encoding an entity.

3.1 Vocabulary Construction

We define the vocabulary of our model as $\mathcal{V} = \mathcal{A} \cup \mathcal{R}_{train}$, where $\mathcal{A}$ is the anchor set constructed from $\mathcal{E}_{train}$ such that $\mathcal{A} \subset \mathcal{E}_{train}$ and $|\mathcal{A}| \ll |\mathcal{E}_{train}|$. This anchor set construction is done by the anchor selection algorithm which selects a set of relationally diverse nodes such that maximum relations are covered in the given anchor size budget. In each iteration we greedily pick anchors with maximum outgoing relation count. For each of those outgoing relations we then remove that relation from the set of outgoing relations for all other nodes that contain that relation. This process is repeated until m anchors are found. Please refer to Algorithm 1 to know the exact steps.

3.2 Anchor Selection per Entity

Given an entity, top m anchors are selected from the anchor set $\mathcal{A}$ based on a relation similarity score. We define two different metrics to capture the relation similarity between a given entity and anchor entity.

Let $\mathcal{S}_e$ be the set of all outgoing relations from the given entity e, and $\mathcal{S}_a$ be the set of all outgoing relations from a candidate anchor entity a.

The first metric is designed to capture the maximum relational overlap between the two sets. For this we use the Jaccard similarity metric given by

$$rsim_{jaccard}(e, a) = \frac{|\mathcal{S}_e \cap \mathcal{S}_a|}{|\mathcal{S}_e \cup \mathcal{S}_a|} \tag{1}$$

This measure for relation similarity uniformly weighs all the relations. Such a metric should be useful in graphs with uniformly distributed relations where the characteristic of a node is not determined by any unique relation. However, in denser relation rich graphs not all relations should receive equal importance in similarity computation. Rarer relations are more informative than the common ones. Drawing inspiration from tf-idf based query document retrieval in NLP, we define the second measure for relation similarity that factors in the frequency of relations

$$rsim_{irf}(e, a) = \frac{1}{|\mathcal{S}_e \cup \mathcal{S}_a|} \sum_{r \in \mathcal{S}_e \cap \mathcal{S}_a} log\left(\frac{|\mathcal{T}_{train}|}{n(r)}\right) \tag{2}$$

where, $\mathcal{T}_{train}$ is the training triples set and $n(r)$ denotes frequency of relation r. The inverse relation frequency based scoring function $rsim_{irf}$ considers the relative frequency of relations in the training graph. In computation of the relation similarity score rarer relations are weighted higher than the frequent ones.

Finally, to select anchors per entity, we select m anchors $\{a_i\}^m \subseteq \mathcal{A}$ with the highest $rsim$ scores.

3.3 Relational Paths per Entity

To create a distinct hash for an entity we represent an entity as a function of n relational paths originating from that entity. We define a k length relational path between two entities e and e_t, as a sequence of k consecutive relations $[r_1, r_2, ..., r_k]$ that are encountered in the shortest path connecting e to e_t . The set of all such relational paths originating from e, denoted by $\mathcal{P}(e)$, is constructed by doing a breadth first search (BFS) starting from e to all entities present in a k hop radius. Finally, n relational paths are selected $\{p_j\}^n \subseteq \mathcal{P}(e)$ following a deterministic or random strategy. Under the deterministic strategy we select n relational paths ordered by their path lengths in non-decreasing order.

3.4 Entity Encoder

Based on the aforementioned tokenization scheme an entity e is tokenized as follows:

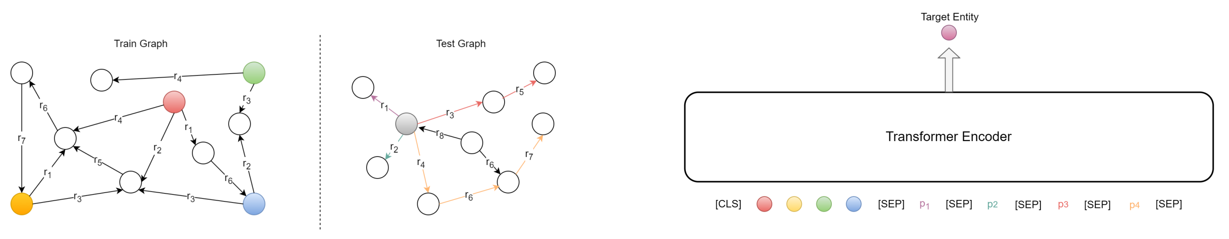
$$hash(e) = [\{a_i\}^m, \{p_j\}^n] \tag{3}$$

The generated hash is further formatted by augmenting special tokens to make it suitable for input to our encoder model.

$$x = [[cls], \{a_i\}^m, \{[sep]\, p_j\}^n] \tag{4}$$

The input tensor $\mathbf{x}$ is then obtained from the embedding lookup table for elements in the vocab, $\mathbf{V} \in \mathbb{R}^{|\mathcal{V}| \times d}$ for each token in the input sequence x.

$$\mathbf{x} = \left[\mathbf{e}_{cls}, \{\mathbf{e}_{a_i}\}^k, \{\mathbf{e}_{sep}\, \mathbf{t}_{p_j}\}^n\right] \tag{5}$$



(a) Anchor selection and relational paths for a test entity

(b) Transformer Encoder

Figure 2: **Entity tokenization and encoding process.** For an unseen test node (in grey), 4 anchors (colored nodes) are selected from the training graph based on maximum relational overlap, and 4 relational paths originating from the test node are sampled to represent the entity. This sequence is then passed through the transformer encoder to obtain the node feature.

Note that, each of the relational paths p_j is a sequence of relations of length k. For relational paths having length less than k we pad them using the pad token. The path embedding $\mathbf{t}_{p_j}$ for a path p_j is thus defined as:

$$\mathbf{t}_{p_j} = \left[\mathbf{e}^{(j)}_{r_1}, \mathbf{e}^{(j)}_{r_2}, \ldots \mathbf{e}^{(j)}_{r_k}\right] \tag{6}$$

We also add the relative positional encodings to the relation embeddings to preserve the sequential information in relational paths.

$$\mathbf{t}_{p_j}(i) = \mathbf{e}^{(j)}_{r_i} + pos_i \tag{7}$$

Finally, we pass the input through L layers of transformer encoder and take the $[cls]$ token embedding as the entity's embedding.

$$\mathbf{x}^l = \text{Transformer}(\mathbf{x}^{l-1}) \tag{8}$$

4 Experiments

The objective of our experimental study is to answer the following research questions:

- How effective is our anchor selection strategy for inductive link prediction?
- Is path based tokenization better than immediate relation based tokenization?
- Does our entity encoder learn meaningful representations?

4.1 Dataset

We evaluate our proposed approach using the benchmark for inductive link prediction introduced by Teru et al. (2020). It comprises of 3 different KG datasets, FB15k-237, WN18RR and NELL-995. Each dataset has 4 different versions that differ in the number of entities, relations and total number of triples. While FB15k-237 and NELL-995 are denser and relationally rich graphs, WN18RR is a sparser graph with fewer relations. For the exact statistics of the dataset we refer the reader to Galkin et al. (2022).

4.2 Setup

The datasets introduced in the inductive benchmark have test graphs which are completely disconnected from train graph. In such settings node connectivity based anchor tokenization is not possible. We employ our proposed approach, where we tokenize an entity into m relationally similar anchors and n relational paths. Subsequently, we pass the tokenized input through a transformer encoder followed by a relationally message passing GNN: CompGCN (Vashishth et al., 2020) to obtain the entity feature. Since our objective is to study the effect of the proposed entity encoding scheme, we use the same RotatE decoder (Sun et al., 2019) used in Nodepiece (Galkin et al., 2022) as the triple scoring function. To train the model we use self-adversarial negative sampling loss (NSSAL). We implement our entity tokenization approach and encoder model on top of NodePiece code. All our models have been trained on A100 GPUs with 40 GB RAM.

4.3 Hyperparameters

We adopt the hyperparameters of the best performing models on inductive benchmark from Nodepiece. On top of that we tune the hyperparameters for our encoder model. For each dataset version we limit the number of anchors $|\mathcal{A}|$ to upto 20% of the total number of entities in the training set. The number of anchors per node m is selected from the following set $\{10, 15, 20, 25\}$. To select the anchor set per node we consider two different similarity measures $rsim_{jaccard}(.)$ and $rsim_{irf}(.)$. In

Method	FB15k-237				WN18RR				NELL-995			
	V1	V2	V3	V4	V1	V2	V3	V4	V1	V2	V3	V4
Neural LP	0.529	0.589	0.529	.0559	0.744	0.689	0.462	0.671	0.408	0.787	0.827	0.806
DRUM	0.529	0.587	0.529	0.559	0.744	0.689	0.462	0.671	0.194	0.786	0.827	0.806
RuleN	0.498	0.778	0.877	0.856	0.809	0.782	0.534	0.716	0.535	0.818	0.773	0.614
GraIL	0.642	0.818	0.828	0.893	0.825	0.787	0.584	0.734	0.595	0.933	0.914	0.732
CoMPILE	0.676	0.829	0.846	0.874	0.836	0.798	0.606	0.754	0.583	0.938	0.927	0.751
TACT	0.657	0.835	0.852	0.886	0.840	0.816	0.679	0.775	0.798	0.889	0.94	0.738
ConGLR	0.682	0.859	0.886	0.893	0.856	**0.929**	0.707	**0.929**	0.81	**0.949**	0.943	0.816
NodePiece	0.873	0.939	0.944	0.949	0.830	0.886	0.785	0.807	0.890	0.901	0.936	0.893
MH-RARe (anchors + 1hop rel)	0.891	0.956	0.958	0.964	0.838	0.883	0.798	0.816	**0.94**	0.918	0.961	0.886
MH-RARe (only path)	0.895	0.953	**0.963**	0.96	**0.867**	0.885	0.8	0.84	0.92	0.905	0.943	0.895
MH-RARe (anchors + path)	**0.905**	**0.961**	0.96	**0.969**	**0.867**	0.893	**0.809**	0.832	0.935	0.926	**0.97**	**0.9**

Table 1: Hits@10 (%) of link prediction for KGs in the inductive setting. Results of baselines are taken from Galkin et. al. (2021). Best results are in **bold**, second best have been underlined.

Dataset	Entity	Selected Anchors
FB15k237-v1	Sense and Sensibility (film)	Atlantic City (1980 film), Brokeback Mountain, Magnolia (film), Gandhi (film), Chicago (2002 film)
	Stoke City F.C.	Celtic F.C., New Zealand national football team, Sunderland A.F.C.
	University of Connecticut	De La Salle University, University of Notre Dame, Boston College
	Rocky V	Rocky, Eagle Eye, American Gangster (film), Assassins (1995 film), Raging Bull
	James Whitmore	Herbert Ross, John C. Reilly, Ben Stiller, Gene Hackman, Jason Robards
NELL-v1	ceo:jeff_bezos	ceo:david_sifry, ceo:andrea_jung, ceo:jeffrey_katzenberg
	televisionstation:wwbt	televisionstation:wpxi_tv,televisionstation:king_tv,website:wbir_tv,televisionstation:kwqc_tv,televisionstation:wcmh_tv
	county:chicago	county:san_francisco,county:philadelphia,city:princeton,city:dallas
	coach:eric_mangini	coach:dennis_green,coach:frank_beamer,coach:bob_stoops,athlete:rick_pitino
	sportsteam:falcons	sportsteam:florida_gators, sportsteam:usc, sportsteam:ravens, sportsteam:new_england_patriots, sportsteam:tennessee_volunteers

Table 2: **Top selected anchors per entity.** Anchors are selected based on the relation similarity score. We see that the selected anchors for an entity are type consistent.

general we find $rsim_{irf}(.)$ to perform better for datasets having triples per entity greater than 2. For a given entity we sample relational paths upto a maximum of 5 hop length. We limit the maximum number of paths for an entity to 20. The hyper-parameters of our best performing models can be found in the Appendix in detail.

4.4 Baselines

We compare our approach MH-RARe against 3 different types of methods for inductive setting, rule-based: Neural LP (Yang et al., 2017), DRUM (Sadeghian et al., 2019), RuleN (Meilicke et al., 2018b), 4 different GNN-based methods: GraIL (Teru et al., 2020), CoMPILE (Mai et al., 2020), TACT (Chen et al., 2021), ConGLR (Lin et al., 2022) and node-tokenization based: NodePiece (Galkin et al., 2022).

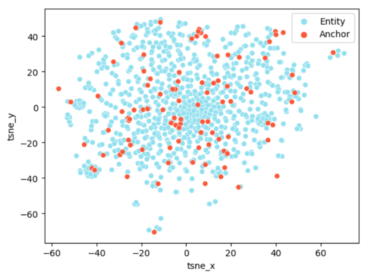
5 Results

5.1 Discussion

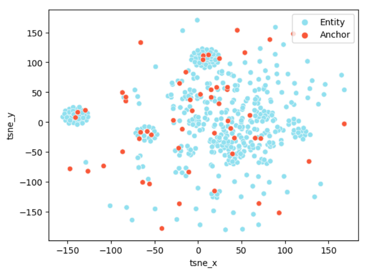
Looking at the general trend of results in Table 1, we observe that anchors are more effective in relationally richer graphs. This is due to the strong inductive bias in our anchor selection strategy which is based on relation similarity. A graph with more relations is likely to have diverse set of relations per entity, which aids in the selection of unique anchors per entity. To complement the performance of anchor based methods on sparse graphs we combine it with path based method. Our path based approach uses multi-hop relational sequence to generate the hash for an entity. The effectiveness of path based approach is evident in the sparser WN18RR graph. Note that, our path based approach is a generalization of 1-hop relation based tokenization proposed in NodePiece. So in dataset versions where multi-hop relational context is not useful, our model retains the performance of Nodepiece. Overall, we find combination of anchor based and relational path based approach to work well attaining an average boost of 2.3%, 2.5% and 2.9% in Hits@10 in FB15k-237, WN18RR and NELL respectively.

5.2 Qualitative Analysis

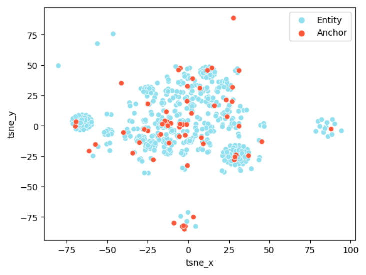
Anchor Selection. On probing the selected anchors per node (Table 2) we find that in general the anchors are type consistent. For e.g. for test node `falcon` we extract `florida gators`, `usc`, `ravens` as anchors, which are all football teams. In a few cases we also observe some anchors which are related but not exactly of the same type for ex. `county:chicago` has anchors which are cities. There are certain cases where a few nodes end up getting noisy anchors. We find this is the case es-



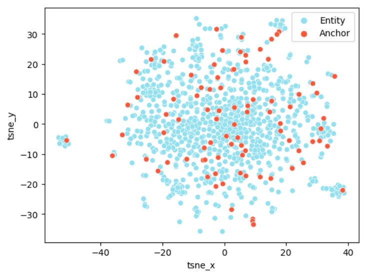
(a) FB15k237-v1 (train)



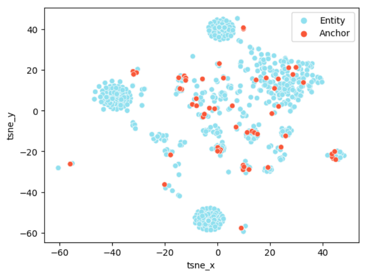
(b) WN18RR-v1 (train)



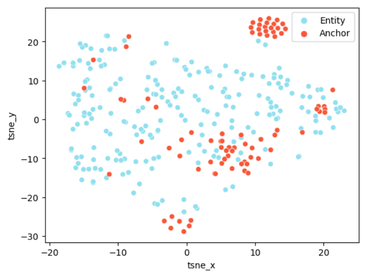
(c) NELL-995-v1 (train)



(d) FB15k237-v1 (test)



(e) WN18RR-v1 (test)



(f) NELL-995 (test)

Figure 3: tSNE projections of train and test entities and their 100 most common anchors

pecially with the nodes that have scarce outdegree and non informative relations. We believe that handling such cases would require anchor selection based on a multi-hop relational context, we leave that upto future work.

Anchor Distribution. To study the quality of the learned embeddings we plot the tSNE projections of entities and anchors in train and test graphs Figure 3. For this we randomly sample 1000 entities from train and test sets of the version 1 of all 3 datasets. If the number of entities are less than 1000 we take all entities. We then compute the top-100 most common anchors across all sampled entities. The embeddings for all sampled entities and their anchors are then obtained from our learned encoder.

tSNE projections of entities and anchors reveal that anchors are evenly distributed across entity clusters, a similar observation was made by Galkin et al. (2022) for transductive KGs. While this trend is evident in the training graphs across all 3 datasets, in test graphs we find that in WN18RR there are some entity clusters not covered by anchors. This provides some evidence towards the performance drop as seen in WN18RR.

6 Conlusion & Future Work

In this paper we have introduced an entity tokenization strategy that tokenizes an entity into anchors and relational paths for inductive KGs. In spite of test graph being disconnected to the training graph, our approach is able to select informative anchors for an unseen entity. We show that while anchors are more effective in relation rich graphs, relational paths can compensate for it's performance in sparser graphs. Overall, addition of anchors leads to an increase in parameter budget, but the number of parameters only grows sub-linearly as the anchor set contains only a small set of representative entities derived from the full entity set.

An interesting direction for future research would be to explore anchor selection in relationally sparse KGs.

References

Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Jiajun Chen, Huarui He, Feng Wu, and Jie Wang. 2021. Topology-aware correlations between relations for inductive link prediction in knowledge graphs. *CoRR*, abs/2103.03642.

William W. Cohen. 2016. Tensorlog: A differentiable deductive database.

Jun Feng, Minlie Huang, Yang Yang, and Xiaoyan Zhu. 2016. GAKE: Graph aware knowledge embedding. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 641–651, Osaka, Japan. The COLING 2016 Organizing Committee.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 413–422, New York, NY, USA. Association for Computing Machinery.

Mikhail Galkin, Etienne Denis, Jiapeng Wu, and William L. Hamilton. 2022. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. In *International Conference on Learning Representations*.

Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1802–1808. AAAI Press.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA. Curran Associates Inc.

Qika Lin, Jun Liu, Fangzhi Xu, Yudai Pan, Yifan Zhu, Lingling Zhang, and Tianzhe Zhao. 2022. Incorporating context graph with logical reasoning for inductive relation prediction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 893–903, New York, NY, USA. Association for Computing Machinery.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1).

Sijie Mai, Shuangjia Zheng, Yuedong Yang, and Haifeng Hu. 2020. Communicative message passing for inductive relation reasoning. *CoRR*, abs/2012.08911.

Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. 2018a. Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion. In *The Semantic Web – ISWC 2018*, pages 3–20, Cham. Springer International Publishing.

Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. 2018b. Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion. In *The Semantic Web – ISWC 2018*, pages 3–20, Cham. Springer International Publishing.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Komal K. Teru, Etienne Denis, and William L. Hamilton. 2020. Inductive relation prediction by subgraph reasoning. *arXiv: Learning*.

Théo Trouillon, Christopher R. Dance, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2017. Knowledge graph completion via complex tensor factorization.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

Peifeng Wang, Jialong Han, Chenliang Li, and Rong Pan. 2019. Logic attention based neighborhood aggregation for inductive knowledge graph embedding. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014, 2020. Knowledge graph embedding by translating on hyperplanes. 28(1).

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases.

Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning.

Dataset		Relations	Train			Validation			Test		
			Entity	Query	Facts	Entity	Query	Facts	Entity	Query	Facts
FB15k-237	v1	183	2,000	4,245	4,245	1,500	206	1,993	1,500	205	1,993
	v2	203	3,000	9,739	9,739	2,000	469	4,145	2,000	478	4,145
	v3	218	4,000	17,986	17,986	3,000	866	7,406	3,000	865	7,406
	v4	222	5,000	27,203	27,203	3,500	1,416	11,714	3,500	1,424	11,714
WN18RR	v1	9	2,746	5,410	5,410	922	185	1,618	922	188	1,618
	v2	10	6,954	15,262	15,262	2,923	411	4,011	2,923	441	4,011
	v3	11	12,078	25,901	25,901	5,084	538	6,327	5,084	605	6,327
	v4	9	3,861	7,940	7,940	7,208	1,394	12,334	7,208	1,429	12,334
NELL-995	v1	14	3,103	4,687	4,687	225	101	833	225	100	833
	v2	88	2,564	8,219	8,219	4,937	459	4,586	4,937	476	4,586
	v3	142	4,647	16,393	16,393	4,921	811	8,048	4,921	809	8,048
	v4	77	2,092	7,546	7,546	3,294	716	7,073	3,294	731	7,073

Table 1: Dataset statistics for inductive link prediction

Parameter	FB15k-237	WN18RR	NELL-995
# Anchors $\|\mathcal{A}\|$	100 | 100 | - | 100	100 | 100 | 100 | -	250 | 150 | 200 | 200
# Anchors per node, m	15 | 15 | - | 15	15 | 10 | 20 | -	15 | 15 | 15 | 20
Relation scoring function	jac | irf | - | irf	irf | irf | irf | -	jac | irf | irf | irf
Max hops, k	3 | 3 | 5 | 3	3 | 3 | 1 | 5	1 | 3 rest
Max Paths n	20 | 15 | 20 | 20	20 | 20 | 5 | 20	6 | 20 rest
Batch size	256	256	256
Learning rate	0.0001	0.0001	0.0001
Num negatives	32	32	32
Epochs	2500 | 2000 rest	1000 | 2000 | 500 | 2000	1000 |3000 | 2500 rest
Trf Layers	2	2	2
Trf Attn Heads	4	4	4
CompGCN layers	3	4 | 6 | 6 | 10	8 | 4 | 4 | 4
CompGCN attention	yes	yes	yes
CompGCN dropout	0.1	0.1	0.1
Loss function	NSSAL	NSSAL	NSSAL
Margin	25 | 25 | 15 | 20	5 | 15 | 5 | 20	25 | 25 | 30 | 20

Table 2: Hyperparameters for inductive link prediction experiments. Hyperparametes for 4 different dataset splits are pipe (|) separated