TRIX: A More Expressive Model for Zero-shot Domain Transfer in Knowledge Graphs

Yucheng Zhang
Purdue University
zhan4332@purdue.edu

Beatrice Bevilacqua
Purdue University
bbevilac@purdue.edu

Mikhail Galkin
Intel AI Lab
mikhail.galkin@intel.com

Bruno Ribeiro
Purdue University
ribeirob@purdue.edu

Abstract

Fully inductive knowledge graph models can be trained on multiple domains and subsequently perform zero-shot knowledge graph completion (KGC) in new unseen domains. This is an important capability towards the goal of having foundation models for knowledge graphs. In this work, we introduce a more expressive and capable fully inductive model, dubbed TRIX, which not only yields strictly more expressive triplet embeddings (head entity, relation, tail entity) compared to state-of-the-art methods, but also introduces a new capability: directly handling both entity and relation prediction tasks in inductive settings. Empirically, we show that TRIX outperforms the state-of-the-art fully inductive models in zero-shot entity and relation predictions in new domains, and outperforms large-context LLMs in out-of-domain predictions. The source code is available at https://github.com/yuchengz99/TRIX.

1 Introduction

Fully inductive knowledge graph models can perform zero-shot Knowledge Graph Completion (KGC), which predicts missing facts in entirely new domains that were not part of the training data. This is particularly challenging because these new domains may contain just unseen relation types and new entities [1–3]. Fully inductive models are trained on one or multiple Knowledge Graphs (KG) with a specific set of relations. Previous work has emphasized the importance of double-equivariance [2], that is, equivariance to permutations of both entity ids and relation ids, as a fundamental property that fully inductive models must have to transfer across KG domains. Intuitively, this equivariance allows models to focus on the underlying structural invariances, despite semantic differences and variations in identifiers across different KGs.

However, despite the notable achievements of current fully inductive models, several open challenges remain, including: (1) Limited expressivity of existing methods; (2) Insufficient support for relation prediction tasks; and (3) Underexploration of the abilities of Large Language Models (LLMs) to perform the same tasks. More precisely, existing state-of-the-art fully inductive models, such as ULTRA [1], have expressivity limitations, as we show in Section 4.3, which implies that certain non-isomorphic triplets inevitably get the same representations, and therefore necessarily the same predictions despite their differences. Since increased expressive power tend to translate into better downstream performances in traditional graph tasks [4–7], an open question is whether improving the expressivity of fully inductive models would also improve their empirical performance. Additionally, existing fully inductive models are primarily designed for entity prediction tasks, answering queries such as (?, relation, tail entity) or (head entity, relation, ?), where the goal is to predict the head or tail entity of a given triplet. Consequently, they miss the equally important relation prediction tasks, namely queries such as (head entity, ?, tail entity), where the goal is to predict the missing

Y. Zhang et al., TRIX: A More Expressive Model for Zero-shot Domain Transfer in Knowledge Graphs. *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

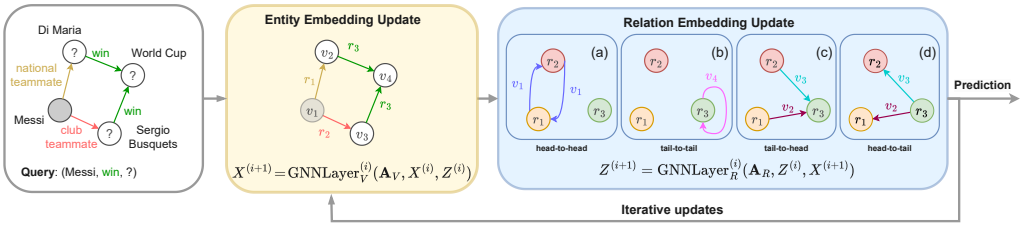


Figure 1: Overview of TRIX showcasing how a KG in a given domain is represented in TRIX's double-equivariant architecture through entity embedding and relation embedding iterative updates.

relation between two entities [8–11]. Finally, while large-context LLMs have demonstrated notable performance in KGC tasks [12–16], their effectiveness in the inductive setting of our interest, where test KGs come from new domains, remains largely underexplored. Therefore, an evaluation and comparison with fully inductive graph models is needed to understand whether fully inductive graph models are necessary or if repurposing LLMs would suffice.

Our approach. In this paper, we aim to address the open challenges in fully inductive models. We first show that the limited expressive power of the state-of-the-art fully inductive models ULTRA [1] arises from its approach in capturing relation interactions, obtained by counting the number of entities sharing a pair of relations, rather than *which* entities share those relations. Then, we propose TRIX (Transferable Relation-Entity Interactions in crossing patterns (X-patterns)), which we show to return *strictly more expressive* triplet representations than existing methods. As illustrated in Figure 1, given any input KG, TRIX first constructs a graph of relations, where each node is a relation from the original graph and edges denote shared entities among those relations. Then, it refines relation and entity representations by iteratively applying graph neural network layers over the graph of relations and the original graph. In this way, TRIX obtains representations of relation and entities directly applicable to zero-shot tasks.

We demonstrate that, by design, TRIX efficiently handles relation prediction tasks, a capability lacking in existing state-of-the-art fully inductive models. In particular, TRIX can answer relation prediction queries in a single forward pass, while existing fully inductive models require performing a number of forward passes equal to the number of relations, for a single relation prediction query.

Finally, we explore the capabilities of LLMs for KGC by designing a comprehensive set of experiments. We demonstrate that, while LLMs can do KGC accurately given enough context about the background knowledge, they rely on the textual information (and its semantics), and therefore fail to utilize the actual graph information, given in the context. We show that this result has several implications, including failure cases when the relation names are not given due to privacy concerns, or simply when they are not known by the LLMs.

Contributions. Our key contributions are as follows: (1) We propose a novel fully inductive model on KGs, TRIX, which exhibits greater expressive power compared to prior methods and can handle relation predictions efficiently; (2) We show that the increased expressiveness of TRIX allows it to surpass state-of-the-art methods in 57 KG datasets in both entity and relation predictions; (3) We present an experimental study of LLMs on the same tasks and show that existing LLMs have limited capabilities in exploiting graph information, which are needed to perform tasks on new domains.

2 Related Work

Fully Inductive Models over KGs. Original efforts in inductive Knowledge Graph Completion primarily focused on handling new entities at test time, but not new relations [8, 17–25]. However, emerging methodologies tackle inductive learning scenarios with both new entities and new relations in test [1–3]. Gao et al. [2] approaches the problem by treating relations as set elements and employing DSS layers [26] to learn representations equivariant to permutations of both node and relation ids. On the other hand, Lee et al. [3] and Galkin et al. [1] introduce relation graphs to capture relation representations based on their interactions. In this work, we extend the latter approach by proposing a novel design for relation graphs, which allows to capture more expressive structural patterns in KGs.

Knowledge Graph Completion with Large Language Models. Despite the impressive capabilities of pretrained LLMs, their application to KGC has primarily focused on leveraging textual information

rather than exploiting the underlying graph structure [13]. Recent efforts aims to enhance LLM performance in KGC by incorporating neighborhood or path information between entities as part of prompts in an in-context learning approach [14–16]. However, these approaches have mainly been applied in transductive settings. Li et al. [27] investigated zero-shot link prediction tasks with LLMs but only considered scenarios involving new relations, not new entities. The effectiveness of LLMs in KGC tasks with both new entities and new relations remains unexplored. Furthermore, the extent to which LLMs exhibit double-equivariance, a crucial property for inductive reasoning on KGs [2], has yet to be thoroughly investigated. In this work, we address these gaps by conducting comprehensive experiments to evaluate the double-equivariance property of LLMs in KGC tasks.

3 Preliminaries

A Knowledge Graph G is a tuple (V, E, R) where V is a finite set of entities, R is a finite set of relations and $E \in (V \times R \times V)$ is a finite set of edges representing relations between entities. We denote by $G_{\text{train}} = (V_{\text{train}}, E_{\text{train}}, R_{\text{train}})$ the training graph, and by $G_{\text{inf}} = (V_{\text{inf}}, E_{\text{inf}}, R_{\text{inf}})$ the test (or inference) graph. Since we focus on inductive settings, where the test graph comes from a different domain, the training and the test graphs have disjoint entity and relation sets, that is, $V_{\text{inf}} \not\subset V_{\text{train}}$ and $R_{\text{inf}} \not\subset R_{\text{train}}$. Despite being unseen, it is however assumed that the test graph shares certain structural patterns with the training graph. These structural similarities imply the existence of invariances sufficient for accurate predictions. Consequently, models trained on the training graph can transfer the learned knowledge to the test graph, leveraging the shared structural patterns for effective predictions.

Since KGs are often incomplete [28], KGC has been widely utilized to infer missing information by predicting missing triplets. KGC comprises two key tasks: entity prediction[29] addresses queries $(h, r, ?)$ which predicts the tail entity given a head entity h and a relation r (or, equivalently, predict the head); relation prediction[8], on the other hand, focuses on queries $(h, ?, t)$, aiming to predict the existence of a link between a head entity h and a tail entity t, and determining the relation type.

To perform entity and relation predictions in inductive settings, Gao et al. [2] recently identified the concept of double-equivariance, i.e., equivariance to both entity and relation ids permutations, as a necessary property that fully inductive models must possess. This property ensures that the architecture does not use relation and entity IDs, but, instead, captures the interactions among them. Indeed, even if relations and entities vary across datasets, the interactions between them may be similar and transferable. Existing fully inductive models achieve double equivariance either by designing architectures that are intrinsically equivariant to both permutation groups [2], or by constructing a relation graph that captures relation interactions regardless of their ids or semantics [1–3]. Since the latter tends to be a lighter approach, which also yields better results in practice, we will in the following focus on that. In particular, these methods first construct a relation graph where entities are relations and edges represent the number of times two relations share an entity. This relation graph is then passed to a standard graph neural network with labeling trick [30] to obtain relation representations conditioned on the query of interest, which can directly be used in inductive settings. We show how TRIX extends this approach and constructs a relation graph that does not merely count the number of entities shared among two relations, but identify which entities are shared, an approach that we show to return strictly more expressive representations in Section 4.3.

Finally, we remark that existing fully inductive models typically begin by applying the labeling trick [30] to obtain initial relative relation representations, conditioned on the query relation [1]. However, in the relation prediction $(h, ?, t)$, there is no specific query relation available for conditioning, as this is unknown and constitutes the target of our query. As a result, existing methods must convert relation prediction into a triplet ranking problem, evaluating the possibility of (h, r, t) for all relations r in the relation set. This implies that the model needs to perform one forward pass for each relation. In contrast, we show in the next section that TRIX can perform the task in one single forward pass.

4 TRIX Framework

As discussed in previous sections, existing fully inductive models suffer from limited expressivity due to the way in which they construct the relation graph and inefficiency in relation prediction tasks due to the message passing scheme. We fill in these gaps by proposing the framework of TRIX, in which the design of our new relation graph records the entity property to enhance expressive power and to the design of iterative message passing mechanism makes relation prediction in one forward pass

possible. Then we explore TRIX's theoretical properties, such as expressiveness and time complexity. An overview of TRIX is illustrated in Figure 1.

4.1 Relation Adjacency Matrix

Existing methods have limited expressive power. For instance, the relation graphs of both ULTRA [1] and InGram [3] are too invariant: In Figure 6 we show how the edges of the relation graphs of these existing methods (how many common entities two relations have) are not expressive enough for the entity prediction task. We increase expressiveness by including entity information, that is, edges between two relations contain information of which entities share these relations. Given the graph G, with adjacency matrix $\boldsymbol{A}_V \in \mathbb{R}^{|V|\times|V|\times|R|}$ (entity adjacency matrix), TRIX first constructs a relation graph G_R with adjacency matrix $\boldsymbol{A}_R$ (relation adjacency matrix). Entities in $\boldsymbol{A}_R$ represent the relations in $\boldsymbol{A}_V$, and edges in $\boldsymbol{A}_R$ denote entities (in $\boldsymbol{A}_V$) that share two relations (in $\boldsymbol{A}_V$). Specifically, for any pair of relations $r_i, r_j \in R$ in the original graph G, we count how many times each entity $v_k \in V$ is part of triplets involving these two relations as: (1) the head entity in both (that is, how many triplets like $(v_k, r_i, \star)$ and $(v_k, r_j, \star)$ exist, where $\star$ is a placeholder for entities), (2) the tail entity in both (that is $(\star, r_i, v_k)$ and $(\star, r_j, v_k)$), (3) the head in the first and the tail in the other (that is $(v_k, r_i, \star)$ and $(\star, r_j, v_k)$), or (4) vice-versa (that is $(\star, r_i, v_k)$ and $(v_k, r_j, \star)$).

Since we keep these four roles (head-head, tail-tail, head-tail, tail-head) separate, and we count the above for each entity $v \in V$, the relation adjacency matrix $\boldsymbol{A}_R$ is a tensor of shape $|R| \times |R| \times |V| \times 4$. Note that this is in contrast with previous methods [1, 3], that, despite also maintaining the distinct roles, do not differentiate which entities participate in the role but only how many, a choice that impacts the expressive power as we shall see next.

Mathematically, we first construct two matrices $\boldsymbol{E}_h \in \mathbb{R}^{|V|\times|R|}$ and $\boldsymbol{E}_t \in \mathbb{R}^{|V|\times|R|}$ capturing how many times each entity $v \in V$ is the head of triplets involving relation $r \in R$, or the tail, respectively. Then, we construct four different intermediate relation adjacency matrices capturing the four roles $\boldsymbol{A}_R^{hh}$, $\boldsymbol{A}_R^{tt}$, $\boldsymbol{A}_R^{ht}$, and $\boldsymbol{A}_R^{th} \in \mathbb{R}^{|R|\times|R|\times|V|}$. These can be directly obtained by leveraging $\boldsymbol{E}_h$ and $\boldsymbol{E}_t$. For instance, the entry $\boldsymbol{A}_R^{hh}[r_i, r_j, v_k]$, which counts how many times $v_k \in V$ is part of triplets involving both relation $r_i \in R$ and relation $r_j \in R$ while being the head entity in both, can be obtained by multiplying entries in $\boldsymbol{E}_h$ as follows:

$$\boldsymbol{A}_R^{hh}[r_i, r_j, v_k] = \boldsymbol{E}_h[v_k, r_i] * \boldsymbol{E}_h[v_k, r_j]. \tag{1}$$

The equations for $\boldsymbol{A}_R^{tt}$, $\boldsymbol{A}_R^{ht}$, $\boldsymbol{A}_R^{th}$ are obtained equivalently by substituting $\boldsymbol{E}_h$ with $\boldsymbol{E}_t$ appropriately and we refer the reader to Appendix B for explicit definitions. These four intermediate adjacency relations are then stacked along the last dimensions, yielding a single $\boldsymbol{A}_R \in \mathbb{R}^{|R|\times|R|\times|V|\times 4}$. Figure 1 ((a), (b), (c) and (d)) contains an illustrative example of these four matrices. Finally, we remark that, although the shape $|R| \times |R| \times |V| \times 4$ of $\boldsymbol{A}_R$ might look massive, it is in fact just an additional list of edge attributes and takes negligible additional space when relying on sparse matrix representations.

4.2 Iterative Entity and Relation Embedding Updates

Existing fully inductive models [1, 3] sequentially perform message passing on the relation adjacency matrix to derive relation representations and subsequently on the entity adjacency matrix using the derived relation representations. As mentioned in Section 3, this sequential message passing does not align with the labeling trick and leads to inefficiency in relation prediction task. We propose a simultaneous refinement process through iterative updates which aligns well with labeling tricks of both relation and entity prediction task. With the more informative relation adjacency matrix A_R proposed in the last part, this iterative embedding update scheme also makes full use of the entity information in A_R to generate strictly more expressive triplet embeddings. Specifically, we perform message passing updates on $\boldsymbol{A}_R$, employing the entity representations as relation embeddings (since entities in $\boldsymbol{A}_V$ correspond to relations in $\boldsymbol{A}_R$). Subsequently, we proceed with message passing layers on $\boldsymbol{A}_V$, utilizing the recently updated relation representations as the relation embeddings. This iterative process is repeated multiple times, ensuring a cohesive refinement of both relation and entity representations throughout the procedure.

In this subsection, we describe mathematically the iterative updates on the entity adjacency matrix and the relation adjacency matrix for the two tasks of interest, namely entity and relation predictions. TRIX is not jointly trained on both tasks, but the proposed framework can be adapted to either solve

entity prediction or relation prediction tasks with different initial embeddings. The objective functions for optimizing the model for these tasks are in Appendix F.1.

Let $\boldsymbol{X}^{(i)} \in \mathbb{R}^{|V| \times d}$ and $\boldsymbol{Z}^{(i)} \in \mathbb{R}^{|R| \times d}$ denote the entity representations and the relation representations of dimension d at any given layer i. Following previous methods [1, 3], we use NBFNet layers with labeling tricks [30] to obtain relative representations conditioned to the query of interest. Therefore, we will use subscripts to denote the conditioning set. For example, $\boldsymbol{X}^{(i)}_{h,r}$ and $\boldsymbol{Z}^{(i)}_{h,r}$ will denote the entity and relation representations conditioned on entity h and relation r. With a slight abuse of notation, we will then refer to the relative representation of entity u conditioned on entity h and relation r as $\boldsymbol{X}^{(i)}_{h,r}(u)$, and, similarly, $\boldsymbol{Z}^{(i)}_{h,r}(r')$ will denote the relative representation of relation r' conditioned on entity h and relation r.

Embeddings for Entity Prediction Tasks. For an entity prediction query $(h, r, ?)$, we leverage the labeling trick and initialize the embeddings of entities and relations to be conditioned on h and r as:

$$\boldsymbol{X}^{(0)}_{h,r}(u) = \text{INIT}_V(h, u), \qquad \boldsymbol{Z}^{(0)}_{h,r}(r') = \text{INIT}_R(r, r') \tag{2}$$

where INIT is the initialization function for embeddings. The initial embeddings for entity h and for relation r are set to all-one vectors, while all the other entities and relations receive initial zero vectors. Subsequently, we perform iterative updates as follows:

$$\begin{aligned}\boldsymbol{X}^{(i)}_{h,r}(u) &= \text{GNNLayer}^{(i)}_V(\boldsymbol{A}_V, \boldsymbol{X}^{(i-1)}_{h,r}, \boldsymbol{Z}^{(i-1)}_{h,r}, u) \\ &= \text{UP}^{(i)}_V\left(\boldsymbol{X}^{(i-1)}_{h,r}(u), \text{AGG}^{(i)}_V\left(\text{MSG}^{(i)}_V(\boldsymbol{X}^{(i-1)}_{h,r}(v), \boldsymbol{Z}^{(i-1)}_{h,r}(r'))|(u, r', v) \in \boldsymbol{A}_V\right)\right) \quad (3)\\ \boldsymbol{Z}^{(i)}_{h,r}(r') &= \text{GNNLayer}^{(i)}_R(\boldsymbol{A}_R, \boldsymbol{Z}^{(i-1)}_{h,r}, \boldsymbol{X}^{(i)}_{h,r}, r') \\ &= \text{UP}^{(i)}_R\left(\boldsymbol{Z}^{(i-1)}_{h,r}(r'), \text{AGG}^{(i)}_R\left(\text{MSG}^{(i)}_R(\boldsymbol{Z}^{(i-1)}_{h,r}(r''), \boldsymbol{X}^{(i)}_{h,r}(u))|(r', u, r'') \in \boldsymbol{A}_R\right)\right) \quad (4)\end{aligned}$$

$\text{GNNLayer}^{(i)}_V$ and $\text{GNNLayer}^{(i)}_R$ are NBFNet layers where $\text{UP}^{(i)}$, $\text{AGG}^{(i)}$ and $\text{MSG}^{(i)}$ stand for update, aggregation, and message functions at the i-th layer, respectively. Following NBFNet, the message function is DistMult, the aggregation function is sum, and the update function is a multi-layer perceptron. The pseudo code is shown in Algorithm 1 in the Appendix. The final entity embeddings are passed to a multi-layer perceptron for entity prediction.

In knowledge graphs, every relation typically has a corresponding reverse relation. To handle the entity prediction query $(?, r, t)$, we can simply transform it into $(t, r^{-1}, ?)$ by utilizing the reverse relation and treat it as a tail prediction as above.

Embeddings for Relation Prediction Tasks. For a relation prediction query $(h, ?, t)$, we leverage the labeling trick and initialize the embeddings of entities and relations to be conditioned on h and t as follows:

$$\boldsymbol{X}^{(0)}_{h,t}(u) = \text{INIT}(h, t, u), \qquad \boldsymbol{Z}^{(0)}_{h,t}(r') = \mathbf{1}^d, \tag{5}$$

where $\mathbf{1}^d$ is an all-one vector of dimension d. This means that the initial embedding for entity h is set to all-one vectors, the initial embedding for entity t is set to all-minus-one vectors, while all the other entities receive initial zero vectors. Furthermore, the relations are initialized all to all-one vectors. Subsequently, we perform iterative embedding updates as follows. Same as entity prediction, $\text{GNNLayer}^{(i)}_V$ and $\text{GNNLayer}^{(i)}_R$ are NBFNet layers:

$$\begin{aligned}\boldsymbol{Z}^{(i)}_{h,t}(r) &= \text{GNNLayer}^{(i)}_R(\boldsymbol{A}_R, \boldsymbol{Z}^{(i-1)}_{h,t}, \boldsymbol{X}^{(i-1)}_{h,t}, r) \\ &= \text{UP}^{(i)}_R\left(\boldsymbol{Z}^{(i-1)}_{h,t}(r), \text{AGG}^{(i)}_R\left(\text{MSG}^{(i)}_R(\boldsymbol{Z}^{(i-1)}_{h,t}(r'), \boldsymbol{X}^{(i-1)}_{h,t}(u))|(r, u, r') \in \boldsymbol{A}_R\right)\right) \quad (6)\\ \boldsymbol{X}^{(i)}_{h,t}(u) &= \text{GNNLayer}^{(i)}_V(\boldsymbol{A}_V, \boldsymbol{X}^{(i-1)}_{h,t}, \boldsymbol{Z}^{(i)}_{h,t}, u) \\ &= \text{UP}^{(i)}_V\left(\boldsymbol{X}^{(i-1)}_{h,t}(u), \text{AGG}^{(i)}_V\left(\text{MSG}^{(i)}_V(\boldsymbol{X}^{(i-1)}_{h,t}(v), \boldsymbol{Z}^{(i)}_{h,t}(r))|(u, r, v) \in \boldsymbol{A}_V\right)\right) \quad (7)\end{aligned}$$

The pseudo code is shown in Algorithm 2 in the Appendix. The final relation embeddings are passed to a multi-layer perceptron for the prediction.

4.3 TRIX Properties

We conduct a theoretical analysis to compare the expressiveness of TRIX with existing relation-graph based fully-inductive models, namely ULTRA [1], InGram [3] and DEq-InGram [2]. More precisely, we aim to investigate the power of both methods to distinguish non-isomorphic triplets. All proofs can be found in Appendix B. We begin by showing that TRIX is at least as expressive as ULTRA and InGram, since it can distinguish all non-isomorphic triplets that they can distinguish. DEq-InGram is a variant of InGram by applying Monte Carlo sampling in inference [3]. Each sampling of DEq-InGram is actually a forward pass of InGram. So for simplicity in the proof we do not include DEq-InGram.

Lemma 1 (TRIX at least as powerful as ULTRA and InGram)**.** *Any non-isomorphic triplet that can be distinguished by ULTRA or InGram can also be distinguished by TRIX with certain choices of hyperparameters and initial embeddings.*

We prove this by showing ULTRA and InGram are actually special cases of TRIX. That is, there exist choices of hyperparameters such that TRIX can precisely implement ULTRA and InGram respectively to reproduce the message passing process of them and can get the same triplet embeddings. However, the contrary is not true: there exist cases where ULTRA or InGram cannot obtain the same triplet embeddings as TRIX, regardless of hyperparameter or weight choices.

Lemma 2 (TRIX can distinguish triplets ULTRA and InGram cannot)**.** *There exist non-isomorphic triplets that can be distinguished by TRIX but that cannot be distinguished by ULTRA and InGram.*

We prove this by constructing exemplary triplets that are clearly non-isomorphic, and then showing that they can be distinguished by TRIX but can not be distinguished by ULTRA and InGram. Finally, we can combine Lemmas 1 and 2 into Theorem 1.

Theorem 1 (TRIX is more expressive than ULTRA and InGram)**.** *TRIX is strictly more expressive than ULTRA and InGram in distinguishing between non-isomorphic triplets in KGs.*

We remark here that the additional expressive power comes from including which entities share two relations, at the cost of an additional dimension in the relation adjacency matrix. Prior relation graphs [1–3] consider all entities as isomorphic: the feature describing the entities shared by two relations is simply how many entities they have in common. It is not uncommon to have many pairs of relations with similar entity counts when these entities are non-isomorphic. Then the relations incorrectly tend to get similar or even same embedding. TRIX distinguish these relations by recording entities in the additional dimension in the relation graph. Through the proposed iterative updates, non-isomorphic entities get diverse embeddings and non-isomorphic relations also get diverse embeddings, thus the expressive power gets boosted. In the following, we discuss the complexity of TRIX.

Time Complexity. Recall that the entity adjacency matrix $\boldsymbol{A}_V \in \mathbb{R}^{|V|\times|V|\times|R|}$ has $|V|$ entities and $|R|$ relations. Denote by α the maximum number of unique relations one single entity connects to as the head or the tail in $\boldsymbol{A}_V$. Then, the relation adjacency matrix $\boldsymbol{A}_R \in \mathbb{R}^{|R|\times|R|\times|V|\times 4}$ has $|R|$ entities, $|V|$ relations and at most $4|V|\alpha^2$ edges. Assuming there are L rounds of iterative updates and embedding dimension is d, which we consider constant, the time complexity of TRIX is $O(|E| + |V|\alpha^2)$ while the time complexity of ULTRA is $O(|E| + |V| + |R|^2)$ for each forward pass, which results in $O(|E| + |V| + |R|^2)$ for entity prediction and in $O((|E| + |V| + |R|^2)|R|)$ for relation prediction, as it needs to perform one forward pass for each relation in the relation set. We expand on this in Appendix B, where we show that, in practice, the number of edges in the relation graph is much smaller than $4|V|\alpha^2$, which therefore results in a complexity $\sim 10\times$ worse than ULTRA in entity prediction, and $\sim 20\times$ better than ULTRA in relation prediction.

5 Experiments

We perform a comprehensive set of experiments to answer the following questions: (1) How does TRIX compare to state-of-the-art fully inductive models in inductive entity and relation prediction tasks and, in particular, does the increased expressiveness translate into better performance? (2) Can the accuracy of TRIX increase with more data in the pre-training? (3) Can LLMs make inductive inferences based on the structural information in KGs? To be more specific, are LLMs equivariant to permutations of relation and entity ids? And, can they do inductive tasks where the relation semantic is hidden and the model needs to leverage the structural information in the input? In the following, we report our main results and refer to Appendix F for additional experiments, including an ablation study on the importance of the proposed relation adjacency matrix and the iterative updates.

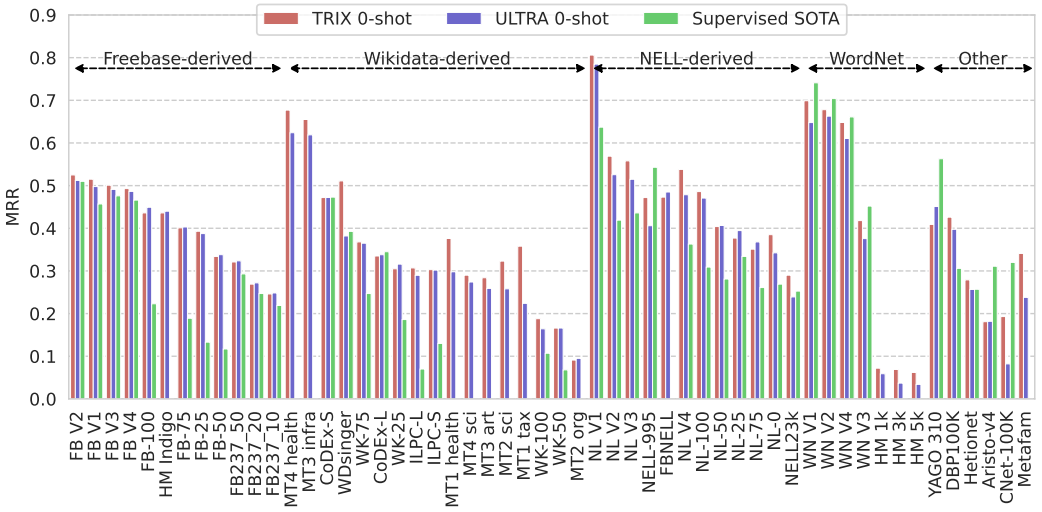


Figure 2: Zero-shot MRR (higher is better) in the entity prediction task. TRIX outperforms ULTRA on 34 datasets, while being comparable on 14 datasets and being outperformed on 6 datasets. It even outperforms supervised baselines on 30 out of 40 datasets.

TRIX Implementation Details. We train one TRIX model for entity prediction and another TRIX model for relation prediction. For relation prediction, TRIX updates relation and entity embedding for 3 rounds, while for entity prediction, for 5 rounds. GNNLayer_R and GNNLayer_V in Equations (3), (4), (6) and (7) are NBFNet layers with hidden dimension 32. In pre-training, the model is trained for 10 epochs with 10000 steps per epoch with early stopping. In fine-tuning, the model is trained for 3 epochs with 1000 steps per epoch with early stopping. We use a batch size of 32, AdamW optimizer and learning rate of 0.0005.

5.1 Zero-Shot Inference and Fine-Tuning of Fully Inductive Models

Datasets & Evaluation. We undertake a comprehensive evaluation across 57 distinct KGs coming from different domains, and split them into three categories: inductive entity and relation (e, r) datasets, inductive entity (e) datasets, and transductive datasets (Appendix E). We follow the procedure prescribed by ULTRA [1] and pretrain on 3 datasets (WN18RR, CoDEx-Medium, FB15k237). We choose ULTRA as our baseline since it is the state-of-the-art double equivariant model that is capable of doing zero-shot fully inductive inference on the large KGs of our interests. For entity prediction task, we report Mean Reciprocal Rank (MRR) and Hits@10 as the main performance metrics evaluated against the full entity set of the inference graph. For each triplet, we report the results of predicting both head and tail. Only in three datasets from Lv et al. [31] we report tail-only metrics, as in the baselines. For relation prediction, we report MRR and Hits@1 as the main performance metrics evaluated against the full relation set of the inference graph. We choose Hits@1 instead of Hits@10 because for some datasets the number of relations is less than 10.

Entity Prediction Results. We present the MRR and Hits@10 results across 57 KGs, along with the corresponding baseline (ULTRA) outcomes in Table 1 and Figure 2. Detailed per-dataset results, including standard deviations, are provided in Tables 16 and 17. In short, TRIX outperforms ULTRA by a significant margin ($\sim 3\%$ average absolute improvement) in zero-shot scenarios, while demonstrating $\sim 0.7\%$ average absolute improvement after fine-tuning in inference tasks. This demonstrates the importance of the additional expressiveness, which results in better performance.

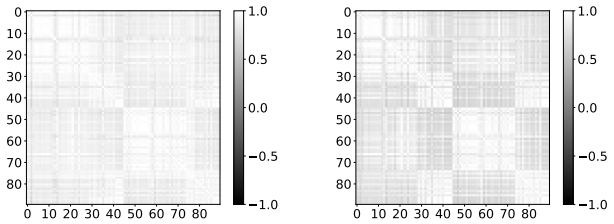
Relation Prediction Results of Graph Models. We present the average MRR and Hits@1 results across 57 KGs, alongside the results for ULTRA in Table 2 (Hits@10 is not chosen because some domains have too few relations). Detailed per-dataset results, including standard deviations, are available in Tables 18 and 19. TRIX outperforms ULTRA significantly in zero-shot inference scenarios, exhibiting substantial advantages, with an average absolute improvement of to 7.4% in Hits@1. Moreover, after fine-tuning, TRIX shows an average absolute improvement of to 4.7% in Hits@1 in inference tasks. Notably, the zero-shot inference results obtained by TRIX even surpass those of ULTRA after finetuning. This underscores the effectiveness of TRIX in relation predictions.

Table 1: Average entity prediction MRR and Hits@10 over 57 KGs from distinct domains. The results over each of the 57 KGs are given in Appendix F.

Model	**Inductive** e, r (23 graphs)		**Inductive** e (18 graphs)		**Transductive** (13 graphs)		**Total Avg** (54 graphs)		**Pretraining** (3 graphs)	
	MRR	**H@10**	**MRR**	**H@10**	**MRR**	**H@10**	**MRR**	**H@10**	**MRR**	**H@10**
ULTRA zero-shot	0.345	0.513	0.431	0.566	0.312	0.458	0.366	0.518	N/A	N/A
TRIX zero-shot	**0.368**	**0.540**	**0.455**	**0.592**	**0.339**	**0.500**	**0.390**	**0.548**	N/A	N/A
ULTRA fine-tuned	0.397	**0.556**	0.442	0.582	0.379	0.543	0.408	0.562	0.407	**0.568**
TRIX fine-tuned	**0.401**	**0.556**	**0.459**	**0.594**	**0.390**	**0.558**	**0.418**	**0.569**	**0.415**	0.563

Table 2: Average relation prediction MRR and hits@1 over 57 KGs from distinct domains. The results over each of the 57 KGs are given in Appendix F.

Model	**Inductive** e, r (23 graphs)		**Inductive** e (18 graphs)		**Transductive** (13 graphs)		**Total Avg** (54 graphs)		**Pretraining** (3 graphs)	
	MRR	**H@1**	**MRR**	**H@1**	**MRR**	**H@1**	**MRR**	**H@1**	**MRR**	**H@1**
ULTRA zero-shot	0.785	0.691	0.714	0.590	0.629	0.507	0.724	0.613	N/A	N/A
TRIX zero-shot	**0.842**	**0.770**	**0.756**	**0.611**	**0.752**	**0.647**	**0.792**	**0.687**	N/A	N/A
ULTRA fine-tuned	0.823	0.741	0.716	0.591	0.707	0.608	0.759	0.659	0.876	**0.817**
TRIX fine-tuned	**0.850**	**0.785**	**0.759**	**0.615**	**0.785**	**0.693**	**0.804**	**0.706**	**0.879**	0.797



(a) Single training domain. **(b)** Four training domains.

Figure 3: Heatmaps of cosine similarities of relation embeddings in test with varying number of training domains. More domains shows stronger embedding differentiation.

Testing Zero-shot Meta-Learning Ability of TRIX. In this experiment, we investigate the meta-learning capability of TRIX. That is, following Thrun and Pratt [32, pp. 4], we can define that a fully inductive model is able to learn-to-learn when increasing the number of learning domains improves the inference performance on new domains. Here, we aim to understand the impact of including more domains in the zero-shot performance of TRIX. We use the WikiTopics dataset [2], which divides the Wikidata-5M [33] into 11 distinct, non-overlapping domains, resulting in 11 unique KGs, each containing different and non overlapping relation and entity sets. We then proceed by training TRIX on a randomly sampled increasing number of domains (until early-stopped) and evaluate their performance on the remaining domains. Table 3 show that as the number of domains in the training increases, the zero-shot inference capability of TRIX improves. Additionally, Figures 3a and 3b illustrate the cosine similarities between the relation embeddings after training, averaged across different query relations, when training with one or four domains, respectively. By adding domains in the training mixture, the model can see more invariances and the relation embeddings get more distinct and expressive, which then translates into more accurate prediction in unseen domains.

5.2 Relation and Entity Prediction with LLMs

To evaluate whether long-context LLMs can make zero-shot relation and entity predictions based on the structural patterns of the graph, we design three different tasks containing both relation prediction and entity prediction. In these tasks, all triplets (head entity, relation, tail entity) of the KG are provided in the prompt. The tasks differ in how entities and relations are represented. The goal of these experiments is to underscore the shortcomings of LLMs in capturing the structural patterns, which is due to being sensitive to permutations of IDs and relying on semantic information.

Experiment Setup. Since all the triplets of KGs are provided in the prompt (approximately 700,000 tokens per query), we selected the Gemini Pro models (Gemini-1.5-flash and Gemini-1.5-pro [34]) as the baseline because they offer the largest token size (1,048,576 tokens for Gemini-1.5-flash), whereas other models, such as GPT-4 (32,000 tokens) and Llama 3.1-70B Instruct (128,000 tokens),

Table 3: TRIX Entity Prediction Hits@10 on WiKiTopics with varying pre-training domains.

# pre-train domains	Average	Art	Award	Edu	Health	Infra	Sci	Sport	Tax
1 domain	0.413	0.380	0.428	0.263	0.601	0.556	0.369	**0.385**	0.324
2 domains	0.459	**0.439**	0.462	0.282	0.700	0.656	0.423	0.366	0.358
3 domains	0.458	0.432	**0.472**	**0.300**	0.623	0.679	0.431	0.376	0.361
4 domains	**0.483**	0.432	0.464	0.294	**0.744**	**0.724**	**0.448**	0.382	**0.396**

have limited token capacities. Due to the cost of the Gemini Pro model, we evaluated on 30 samples from the CoDEx-S dataset [35]. The prompts used followed previous works [14, 36]. We report in the following relation prediction results, while entity prediction results and further experimental details, including prompts, can be found in Appendix D.

Task 1: In-domain LLM predictions. The entities and relations are expressed by their names in natural language, which means that the LLM is queried in the way it was trained on. This task evaluates the in-domain prediction capacity of long-context LLMs. Figure 4 (and Appendix D.1) shows that the LLM handles the in-domain relation prediction nearly as well as TRIX and better than ULTRA.

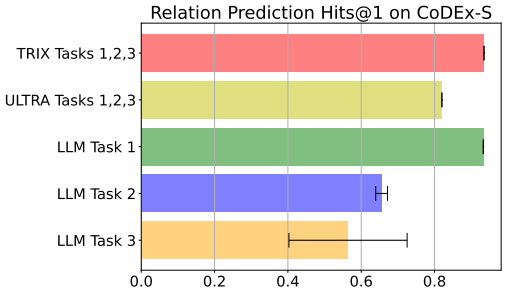


Figure 4: Relation prediction Hits@1 of Gemini-1.5-Pro on three tasks against fully inductive models. Double-equivariant graph models always give consistent performance but LLM performance changes drastically with metasyntactic tokens (Task 2) and ID permutations (Task 3).

Task 2: Out-of-domain LLM predictions. The neighbor entities of the head entity and the relations that connect the head entity with its neighbors are replaced with metasyntactic words [37] like `foo`, `bar` and `baz` across the whole KG to simulate information from a new domain (with new entities and relations) while other entities and relations are still expressed by their names in natural language. This task tests whether long-context LLMs reason on the KG's structural information for the inductive predictions of the metasyntactic words, or if they just simply rely on its pre-trained semantic information. Figure 4 (details in Appendix D.2) shows that with metasyntactic words, Gemini-1.5-pro fails to make consistent accurate predictions. Hence, the good performance in Task 1 likely came from its in-distribution syntax, which means that even SOTA long-context LLMs struggle making relation predictions in new KGs domains using only relational information.

Task 3: Double-equivariant LLM predictions. All the entities and relations are expressed with IDs (e.g. "entity 64", "relation 22") as in Shu et al. [14]. The mappings from entities and relations to their IDs are given to the LLM and the IDs are shuffled in each test run. This task tests whether long-context LLMs are (double) equivariant to entity and relation ID permutations. Figure 4 (details in Appendix D.3) shows that the performance of a long-context LLM changes drastically with the permutation of relation and entity IDs. Only for 38.5% of all queries the model makes consistent predictions across 3 runs, which indicates LLM is quite sensitive to the input permutations of the KG. With the whole graph sent as an edge list, the task resembles long text understanding and retrieval tests akin to "a needle in a haystack" [38–40] where LLMs show increasingly better performance on context lengths up to 128k tokens. However, our results indicate that such (often synthetic) benchmarks might be overestimating real long-context reasoning capabilities of LLMs (let alone adding a simple relation prediction task on top of the input context).

6 Conclusion

In this paper we considered the fully inductive link prediction task in KGs. We identified the open challenges in existing fully inductive models, and proposed TRIX, a novel architecture designed to improve expressiveness and support efficient relation prediction tasks. Through comprehensive experiments spanning 57 diverse KGs datasets, we demonstrate that increased expressiveness translates into better performance. Additionally, our experimental study sheds light on the limitations of LLMs in exploiting graph information in new domains for entity and relation prediction tasks.

Acknowledgments

This work was funded in part by the National Science Foundation (NSF) awards, CCF-1918483, CAREER IIS-1943364 and CNS-2212160, Amazon Research Award, AnalytiXIN, and the Wabash Heartland Innovation Network (WHIN), Ford, NVidia, CISCO, and Amazon. Computing infrastructure was supported in part by CNS-1925001 (CloudBank). This work was supported in part by AMD under the AMD HPC Fund program.

References

[1] Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 2, 3, 4, 5, 6, 7

[2] Jianfei Gao, Yangze Zhou, Jincheng Zhou, and Bruno Ribeiro. Double equivariance for inductive link prediction for both new nodes and new relation types. *arXiv preprint arXiv:2302.01313*, 2023. 1, 2, 3, 6, 8

[3] Jaejun Lee, Chanyoung Chung, and Joyce Jiyoung Whang. Ingram: Inductive knowledge graph embedding via relation graphs. In *Proceedings of the 40th International Conference on Machine Learning*, pages 18796–18809, 2023. 1, 2, 3, 4, 5, 6, 23

[4] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1

[5] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2022.

[6] Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of GNNs via graph biconnectivity. In *The Eleventh International Conference on Learning Representations*, 2023.

[7] Omri Puny, Derek Lim, Bobak Kiani, Haggai Maron, and Yaron Lipman. Equivariant polynomials for graph neural networks. In *International Conference on Machine Learning*, pages 28191–28222. PMLR, 2023. 1

[8] Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR, 2020. 2, 3, 23

[9] Zijun Cui, Pavan Kapanipathi, Kartik Talamadupula, Tian Gao, and Qiang Ji. Type-augmented relation prediction in knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7151–7159, 2021.

[10] Ke Liang, Lingyuan Meng, Sihang Zhou, Wenxuan Tu, Siwei Wang, Yue Liu, Meng Liu, Long Zhao, Xiangjun Dong, and Xinwang Liu. Mines: Message intercommunication for inductive relation reasoning over neighbor-enhanced subgraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10645–10653, 2024.

[11] Zhixiang Su, Di Wang, Chunyan Miao, and Lizhen Cui. Anchoring path for inductive relation prediction in knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9011–9018, 2024. 2

[12] Yichi Zhang, Zhuo Chen, Wen Zhang, and Huajun Chen. Making large language models perform better in knowledge graph completion. *arXiv preprint arXiv:2310.06671*, 2023. 2

[13] Chen Chen, Yufei Wang, Aixin Sun, Bing Li, and Kwok-Yan Lam. Dipping plms sauce: Bridging structure and text for effective knowledge graph completion via conditional soft prompting. *arXiv preprint arXiv:2307.01709*, 2023. 3

[14] Dong Shu, Tianle Chen, Mingyu Jin, Yiting Zhang, Mengnan Du, and Yongfeng Zhang. Knowledge graph large language model (kg-llm) for link prediction. *arXiv preprint arXiv:2403.07311*, 2024. 3, 9

[15] Derong Xu, Ziheng Zhang, Zhenxi Lin, Xian Wu, Zhihong Zhu, Tong Xu, Xiangyu Zhao, Yefeng Zheng, and Enhong Chen. Multi-perspective improvement of knowledge graph completion with large language models. *arXiv preprint arXiv:2403.01972*, 2024.

[16] Yanbin Wei, Qiushi Huang, James T Kwok, and Yu Zhang. Kicgpt: Large language model with knowledge in context for knowledge graph completion. *arXiv preprint arXiv:2402.02389*, 2024. 2, 3

[17] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer, 2018. 2

[18] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34:29476–29490, 2021. 14

[19] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. *Advances in neural information processing systems*, 30, 2017.

[20] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in Neural Information Processing Systems*, 32, 2019.

[21] Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. *arXiv preprint arXiv:2010.04029*, 2020.

[22] Kewei Cheng, Jiahao Liu, Wei Wang, and Yizhou Sun. Rlogic: Recursive logical rule learning from knowledge graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 179–189, 2022.

[23] Chen Shengyuan, Yunfeng Cai, Huang Fang, Xiao Huang, and Mingming Sun. Differentiable neuro-symbolic reasoning on large-scale knowledge graphs. *Advances in Neural Information Processing Systems*, 36, 2024.

[24] Shuwen Liu, Bernardo Grau, Ian Horrocks, and Egor Kostylev. Indigo: Gnn-based inductive knowledge graph completion using pair-wise encoding. *Advances in Neural Information Processing Systems*, 34:2034–2045, 2021. 23

[25] David Jaime Tena Cucala, Bernardo Cuenca Grau, Egor V. Kostylev, and Boris Motik. Explainable GNN-based models over knowledge graphs. In *International Conference on Learning Representations*, 2022. 2

[26] Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On learning sets of symmetric elements. In *International conference on machine learning*, pages 6734–6744. PMLR, 2020. 2

[27] Mingchen Li, Chen Ling, Rui Zhang, and Liang Zhao. A condensed transition graph framework for zero-shot link prediction with large language models. *arXiv preprint arXiv:2402.10779*, 2024. 3

[28] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–49, 2021. 3

[29] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013. 3

[30] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. *Advances in Neural Information Processing Systems*, 34:9061–9073, 2021. 3, 5, 14

[31] Xin Lv, Xu Han, Lei Hou, Juanzi Li, Zhiyuan Liu, Wei Zhang, Yichi Zhang, Hao Kong, and Suhui Wu. Dynamic anticipation and completion for multi-hop reasoning over sparse knowledge graph. *arXiv preprint arXiv:2010.01899*, 2020. 7

[32] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998. 8

[33] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021. 8

[34] Gemini Team, Machel Reid, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. 8

[35] Tara Safavi and Danai Koutra. Codex: A comprehensive knowledge graph completion benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8328–8350. Association for Computational Linguistics, 2020. 9, 22

[36] Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. Exploring large language models for knowledge graph completion. *arXiv preprint arXiv:2308.13916*, 2023. 9

[37] Wikipedia. Metasyntactic variable - wikipedia. https://en.wikipedia.org/wiki/Metasyntactic_variable, 2024. Accessed: date-of-access. 9

[38] Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. In search of needles in a 11m haystack: Recurrent memory finds what llms miss. *arXiv preprint arXiv:2402.10790*, 2024. 9

[39] Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. ZeroSCROLLS: A zero-shot benchmark for long text understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7977–7989. Association for Computational Linguistics, 2023.

[40] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023. 9

[41] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019. 14

[42] Pablo Barceló, Mikhail Galkin, Christopher Morris, and Miguel Romero Orth. Weisfeiler and leman go relational. In *Learning on Graphs Conference*, pages 46–1. PMLR, 2022. 14

[43] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Yannick Hammerla, Michael M Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. In *The eleventh international conference on learning representations*, 2022. 14

[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. 19

[45] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019. 19

[46] Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com. 19

[47] Xin Lv, Xu Han, Lei Hou, Juanzi Li, Zhiyuan Liu, Wei Zhang, Yichi Zhang, Hao Kong, and Suhui Wu. Dynamic anticipation and completion for multi-hop reasoning over sparse knowledge graph. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5694–5703. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.459. 22

[48] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66, 2015. 22

[49] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 22

[50] Yihong Chen, Pasquale Minervini, Sebastian Riedel, and Pontus Stenetorp. Relation prediction as an auxiliary training objective for improving multi-relational graph representations. In *3rd Conference on Automated Knowledge Base Construction*, 2021. 22

[51] Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *Elife*, 6:e26726, 2017. 22

[52] Wenhan Xiong, Thien Hoang, and William Yang Wang. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573. Association for Computational Linguistics, 2017. 22

[53] Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2925–2933, 2020. 22

[54] Boyang Ding, Quan Wang, Bin Wang, and Li Guo. Improving knowledge graph embedding using simple constraints. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 110–121. Association for Computational Linguistics, 2018. 22

[55] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *7th biennial conference on innovative data systems research*. CIDR Conference, 2014. 22

[56] Mikhail Galkin, Max Berrendorf, and Charles Tapley Hoyt. An Open Challenge for Inductive Link Prediction on Knowledge Graphs. *arXiv preprint arXiv:2203.01520*, 2022. 23

[57] 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 Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1802–1808, 2017. doi: 10.24963/ijcai.2017/250. 23

[58] Jincheng Zhou, Beatrice Bevilacqua, and Bruno Ribeiro. An ood multi-task perspective for link prediction with new relation types and nodes. *arXiv preprint arXiv:2307.06046*, 2023. 23

A Pseudo Codes of Iterative Embedding Updates

Algorithm 1 and Algorithm 2 show the iterative embedding updates in entity prediction task and relation prediction task respectively.

Algorithm 1 TRIX embedding updates for the entity prediction task

Require: Query $(h, r, ?)$; relation adjacency matrix $\boldsymbol{A}_R$; entity adjacency matrix $\boldsymbol{A}_V$; number of updates L
Ensure: Final entity embedding $\boldsymbol{X}_{h,r}^{(L)}$

$\boldsymbol{X}_{h,r}^{(0)} = \text{INIT}_V(h)$ ▷ Label h with all-ones vector and the rest with all-zeros
$\boldsymbol{Z}_{h,r}^{(0)} = \text{INIT}_R(r)$ ▷ Label r with all-ones vector and the rest with all-zeros
for $i \leftarrow 1$ to L **do**
  for $u \in V$ **do**
    $\boldsymbol{X}_{h,r}^{(i)}(u) = \text{UP}_V^{(i)}\left(\boldsymbol{X}_{h,r}^{(i-1)}(u), \text{AGG}_V^{(i)}\left(\text{MSG}_V^{(i)}(\boldsymbol{X}_{h,r}^{(i-1)}(v), \boldsymbol{Z}_{h,r}^{(i-1)}(r'))|(u, r', v) \in \boldsymbol{A}_V\right)\right)$
  end for
  for $r' \in R$ **do**
    $\boldsymbol{Z}_{h,r}^{(i)}(r') = \text{UP}_R^{(i)}\left(\boldsymbol{Z}_{h,r}^{(i-1)}(r'), \text{AGG}_R^{(i)}\left(\text{MSG}_R^{(i)}(\boldsymbol{Z}_{h,r}^{(i-1)}(r''), \boldsymbol{X}_{h,r}^{(i)}(u))|(r', u, r'') \in \boldsymbol{A}_R\right)\right)$
  end for
end for

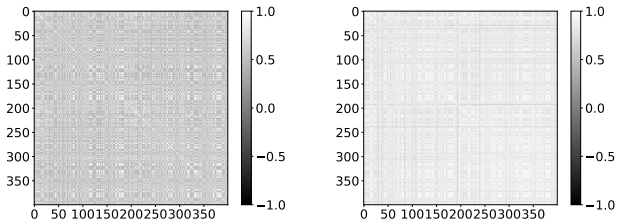
Algorithm 2 TRIX embedding updates for the relation prediction task

Require: Query $(h, ?, t)$; relation adjacency matrix $\boldsymbol{A}_R$; entity adjacency matrix $\boldsymbol{A}_V$; number of updates L
Ensure: Final relation embedding $\boldsymbol{Z}_{h,t}^{(L)}$

$\boldsymbol{Z}_{h,t}^{(0)} = \mathbf{1}^{|R| \times d}$
$\boldsymbol{X}_{h,t}^{(0)} = \text{INIT}(h, t)$ ▷ Label h with all-ones vector, t with all-negative-ones and the rest with all-zeros
for $i \leftarrow 1$ to L **do**
  for $r \in R$ **do**
    $\boldsymbol{Z}_{h,t}^{(i)}(r) = \text{UP}_R^{(i)}\left(\boldsymbol{Z}_{h,t}^{(i-1)}(r), \text{AGG}_R^{(i)}\left(\text{MSG}_R^{(i)}(\boldsymbol{Z}_{h,t}^{(i-1)}(r'), \boldsymbol{X}_{h,t}^{(i-1)}(u))|(r, u, r') \in \boldsymbol{A}_R\right)\right)$
  end for
  for $u \in V$ **do**
    $\boldsymbol{X}_{h,t}^{(i)}(u) = \text{UP}_V^{(i)}\left(\boldsymbol{X}_{h,t}^{(i-1)}(u), \text{AGG}_V^{(i)}\left(\text{MSG}_V^{(i)}(\boldsymbol{X}_{h,t}^{(i-1)}(v), \boldsymbol{Z}_{h,t}^{(i)}(r))|(u, r, v) \in \boldsymbol{A}_V\right)\right)$
  end for
end for

B Expressive Power

The expressive power of a GNN refers to its ability of distinguishing non-isomorphic graphs. Previous works [41] have shown that the expressive power of GNN is bounded by Weisfeiler–Leman tests. Recent works [42] also extend Weisfeiler–Leman tests on relational graphs. The expressive power in link prediction is limited due to the automorphic node problem [43]. Labeling trick [18, 30] breaks the symmetry of entity embedding in link prediction by assigning each node a unique feature vector based on its structural properties.



(a) TRIX. **(b)** ULTRA.

Figure 5: Heatmaps of cosine similarities of relation embeddings on NELL995 with TRIX and ULTRA. TRIX gets more diverse relation embeddings than that of ULTRA.

Section 4.3 shows that TRIX is more expressive than existing relation-graph based double-equivariant models. Before going into the details, we discuss the implications of this expressivity increase, such that the relation embeddings that become more distinguishable. Figure 5 shows the heatmaps of cosine similarity of relation embeddings of ULTRA and TRIX on the NELL995 dataset. The heatmap

of TRIX shows more shade ranges than that of ULTRA, which indicates that the relation embeddings of TRIX are more distinct.

In the following part we discuss the expressive power of TRIX. We start by formally introducing the four intermediate relation adjacency matrices. Specifically, recall that the entity adjacency matrix $\boldsymbol{A}_V \in \mathbb{R}^{|V|\times|V|\times|R|}$ is obtained by stacking four relation adjacency matrices capturing the four roles $\boldsymbol{A}_R^{hh}$ (head-head), $\boldsymbol{A}_R^{tt}$ (tail-tail), $\boldsymbol{A}_R^{ht}$ (head-tail), and $\boldsymbol{A}_R^{th}$ (tail-head). These can be directly obtained by leveraging $\boldsymbol{E}_h$ and $\boldsymbol{E}_t$, that is:

$$\boldsymbol{A}_R^{hh}[r_i, r_j, v_k] = \boldsymbol{E}_h[v_k, r_i] * \boldsymbol{E}_h[v_k, r_j], \qquad \boldsymbol{A}_R^{tt}[r_i, r_j, v_k] = \boldsymbol{E}_t[v_k, r_i] * \boldsymbol{E}_t[v_k, r_j],$$
$$\boldsymbol{A}_R^{ht}[r_i, r_j, v_k] = \boldsymbol{E}_h[v_k, r_i] * \boldsymbol{E}_t[v_k, r_j], \qquad \boldsymbol{A}_R^{th}[r_i, r_j, v_k] = \boldsymbol{E}_t[v_k, r_i] * \boldsymbol{E}_h[v_k, r_j].$$

Next we will discuss the details of the proof.

Lemma 1 (TRIX at least as powerful as ULTRA and InGram)**.** *Any non-isomorphic triplet that can be distinguished by ULTRA or InGram can also be distinguished by TRIX with certain choices of hyperparameters and initial embeddings.*

Proof. In this proof we consider the task of entity prediction.

TRIX is at least as expressive as ULTRA. Suppose ULTRA first updates relation embedding with a k_1-layer GNN and then updates entity embedding with another k_2-layer GNN. We will show that there is a TRIX with $(k_1 + k_2)$ rounds of iterative updates that will get the same embeddings as ULTRA. TRIX uses the same initial relation entity embedding as ULTRA and uses the all-one matrix as initial entity embedding. To be consistent with ULTRA, TRIX uses NBFNet as the GNNLayer.

For the first k_1 rounds, ULTRA only updates relation embedding with message passing on relation graphs. ULTRA's message passing function on relation graph is

$$\boldsymbol{Z}_{h,r}^{(i)}(r') = \mathrm{UP}_{R,\mathrm{ULTRA}}^{(i)}\left(\boldsymbol{Z}_{h,r}^{(i-1)}(r'), \mathrm{AGG}_{R,\mathrm{ULTRA}}^{(i)}\left(\mathrm{MSG}_{R,\mathrm{ULTRA}}^{(i)}(\boldsymbol{Z}_{h,r}^{(i-1)}(r'')|(r', r'') \in \boldsymbol{A}_R^{\mathrm{ULTRA}}\right)\right)$$

For TRIX, we choose the message passing functions on relation graph as follows:

$$\begin{aligned}
\boldsymbol{X}_{h,r}^{(i)}(u) &= \mathrm{UP}_{V,\mathrm{TRIX}}^{(i)}\Big(\boldsymbol{X}_{h,r}^{(i-1)}(u), \\
&\quad \mathrm{AGG}_{V,\mathrm{TRIX}}^{(i)}\left(\mathrm{MSG}_{V,\mathrm{TRIX}}^{(i)}(\boldsymbol{X}_{h,r}^{(i-1)}(v), \boldsymbol{Z}_{h,r}^{(i-1)}(r'))|(u, r', v) \in \boldsymbol{A}_V^{\mathrm{TRIX}}\right)\Big) \\
&= \boldsymbol{X}_{h,r}^{(i-1)}(u) = \mathbf{1}^d \\
\boldsymbol{Z}_{h,r}^{(i)}(r') &= \mathrm{UP}_{R,\mathrm{TRIX}}^{(i)}\Big(\boldsymbol{Z}_{h,r}^{(i-1)}(r'), \\
&\quad \mathrm{AGG}_{R,\mathrm{TRIX}}^{(i)}\left(\mathrm{MSG}_{R,\mathrm{TRIX}}^{(i)}(\boldsymbol{Z}_{h,r}^{(i-1)}(r''), \boldsymbol{X}_{h,r}^{(i)}(u))|(r', u, r'') \in \boldsymbol{A}_R^{\mathrm{TRIX}}\right)\Big)
\end{aligned}$$

Both ULTRA and TRIX use the non-parametric DistMult as the message function. If $\boldsymbol{X}_{h,r}^{(i)}$ is an all-one matrix, we have

$$\begin{aligned}
&\mathrm{MSG}_{R,\mathrm{ULTRA}}^{(i)}(\boldsymbol{Z}_{h,r}^{(i-1)}(r'')|(r', r'') \in \boldsymbol{A}_R^{\mathrm{ULTRA}}) \\
=&\mathrm{MSG}_{R,\mathrm{TRIX}}^{(i)}(\boldsymbol{Z}_{h,r}^{(i-1)}(r''), \boldsymbol{X}_{h,r}^{(i)}(u))|(r', u, r'') \in \boldsymbol{A}_R^{\mathrm{TRIX}})
\end{aligned}$$

Both ULTRA and TRIX use sum as the aggregation function. If $\mathrm{UP}_{R,\mathrm{ULTRA}}^{(i)}$ and $\mathrm{UP}_{R,\mathrm{TRIX}}^{(i)}$ always have the same parameter, it will end up with the same $\boldsymbol{Z}_{h,r}^{(k_1)}$.

For the last k_2 layers, ULTRA only updates entity embedding. ULTRA marks the head entity with the query relation embedding. ULTRA's message passing function on the original knowledge graph is:

$$\begin{aligned}
\boldsymbol{X}_{h,r}^{(i)}(u) &= \mathrm{UP}_{V,\mathrm{ULTRA}}^{(i)}\Big(\boldsymbol{X}_{h,r}^{(i-1)}(u), \\
&\quad \mathrm{AGG}_{V,\mathrm{ULTRA}}^{(i)}\left(\mathrm{MSG}_{V,\mathrm{ULTRA}}^{(i)}(\boldsymbol{X}_{h,r}^{(i-1)}(v), \boldsymbol{Z}_{h,r}^{(k_1)}(r'))|(u, r', v) \in \boldsymbol{A}_V^{\mathrm{ULTRA}}\right)\Big)
\end{aligned}$$

For TRIX, it also marks the head entity with the query relation embedding as ULTRA does. We choose the message passing functions on relation graph as follows:

$$\begin{aligned}
\boldsymbol{X}_{h,r}^{(i+k_1)}(u) &= \mathrm{UP}_{V,\mathrm{TRIX}}^{(i+k_1)}\Big(\boldsymbol{X}_{h,r}^{(i+k_1-1)}(u), \\
&\qquad \mathrm{AGG}_{V,\mathrm{TRIX}}^{(i+k_1)}\Big(\mathrm{MSG}_{V,\mathrm{TRIX}}^{(i+k_1)}(\boldsymbol{X}_{h,r}^{(i+k_1-1)}(v), \boldsymbol{Z}_{h,r}^{(i+k_1-1)}(r'))|(u,r',v) \in \boldsymbol{A}_V^{\mathrm{TRIX}}\Big)\Big) \\
\boldsymbol{Z}_{h,r}^{(i+k_1)}(r') &= \mathrm{UP}_{R,\mathrm{TRIX}}^{(i+k_1)}\Big(\boldsymbol{Z}_{h,r}^{(i+k_1-1)}(r'), \\
&\qquad \mathrm{AGG}_{R,\mathrm{TRIX}}^{(i+k_1)}\Big(\mathrm{MSG}_{R,\mathrm{TRIX}}^{(i+k_1)}(\boldsymbol{Z}_{h,r}^{(i+k_1-1)}(r''), \boldsymbol{X}_{h,r}^{(i+k_1)}(u))|(r',u,r'') \in \boldsymbol{A}_R^{\mathrm{TRIX}}\Big)\Big) \\
&\qquad = \boldsymbol{Z}_{h,r}^{(i+k_1-1)}(r') = \boldsymbol{Z}_{h,r}^{(k_1)}
\end{aligned}$$

If $\mathrm{UP}_{V,\mathrm{ULTRA}}^{(i)}$ and $\mathrm{UP}_{V,\mathrm{TRIX}}^{(i+k_1)}$ always have the same parameter, the relation and entity embedding from ULTRA and TRIX are exactly the same. Since the embeddings are the same, any non-isomorphic triplet that can be distinguished by ULTRA can also be distinguished by TRIX.

TRIX is at least as expressive as InGram. Suppose InGram first updates relation embedding with a k_1-layer GNN and then updates entity embedding with another k_2-layer GNN. We will show that there is a TRIX with $(k_1 + k_2)$ rounds of iterative updates that will get the same embeddings as InGram. TRIX uses the same random initial relation entity embedding as InGram and uses the all-one matrix as initial entity embedding. To be consistent with InGram, TRIX uses an extension of GATv2 as the GNNLayer.

For the first k_1 rounds, InGram only updates relation embedding with message passing on relation graphs. InGram's message passing function on relation graph is

$$\boldsymbol{Z}_{h,r}^{(i)}(r') = \mathrm{UP}_{R,\mathrm{InGram}}^{(i)}\Big(\boldsymbol{Z}_{h,r}^{(i-1)}(r'), \mathrm{AGG}_{R,\mathrm{InGram}}^{(i)}\Big(\mathrm{MSG}_{R,\mathrm{InGram}}^{(i)}(\boldsymbol{Z}_{h,r}^{(i-1)}(r'')|(r',r'') \in \boldsymbol{A}_R^{\mathrm{InGram}}\Big)\Big)$$

For TRIX, we choose the message passing functions on relation graph as follows. Here with a slight abuse of notation, $\boldsymbol{A}_{R,hh}^{\mathrm{TRIX}}$ and $\boldsymbol{A}_{R,tt}^{\mathrm{TRIX}}$ are the relation adjacency matrices for the head-head and tail-tail connections respectively.

$$\begin{aligned}
\boldsymbol{X}_{h,r}^{(i)}(u) &= \mathrm{UP}_{V,\mathrm{TRIX}}^{(i)}\Big(\boldsymbol{X}_{h,r}^{(i-1)}(u), \\
&\qquad \mathrm{AGG}_{V,\mathrm{TRIX}}^{(i)}\Big(\mathrm{MSG}_{V,\mathrm{TRIX}}^{(i)}(\boldsymbol{X}_{h,r}^{(i-1)}(v), \boldsymbol{Z}_{h,r}^{(i-1)}(r'))|(u,r',v) \in \boldsymbol{A}_V^{\mathrm{TRIX}}\Big)\Big) \\
&\qquad = \boldsymbol{X}_{h,r}^{(i-1)}(u) = \mathbf{1}^d \\
\boldsymbol{Z}_{h,r}^{(i)}(r') &= \mathrm{UP}_{R,\mathrm{TRIX}}^{(i)}\Big(\boldsymbol{Z}_{h,r}^{(i-1)}(r'), \\
&\qquad \mathrm{AGG}_{R,\mathrm{TRIX}}^{(i)}\Big(\mathrm{MSG}_{R,\mathrm{TRIX}}^{(i)}(\boldsymbol{Z}_{h,r}^{(i-1)}(r''), \boldsymbol{X}_{h,r}^{(i)}(u))|(r',u,r'') \in \big(\boldsymbol{A}_{R,hh}^{\mathrm{TRIX}} \cup \boldsymbol{A}_{R,tt}^{\mathrm{TRIX}}\big)\Big)\Big)
\end{aligned}$$

TRIX picks the same message function as InGram. If $\boldsymbol{X}_{h,r}^{(i)}$ is an all-one matrix, we have

$$\begin{aligned}
&\mathrm{MSG}_{R,\mathrm{InGram}}^{(i)}(\boldsymbol{Z}_{h,r}^{(i-1)}(r'')|(r',r'') \in \boldsymbol{A}_R^{\mathrm{InGram}}) \\
=&\mathrm{MSG}_{R,\mathrm{TRIX}}^{(i)}(\boldsymbol{Z}_{h,r}^{(i-1)}(r''), \boldsymbol{X}_{h,r}^{(i)}(u))|(r',u,r'') \in \big(\boldsymbol{A}_{R,hh}^{\mathrm{TRIX}} \cup \boldsymbol{A}_{R,tt}^{\mathrm{TRIX}}\big))
\end{aligned}$$

Both InGram and TRIX use sum as the aggregation function. If $\mathrm{UP}_{R,\mathrm{InGram}}^{(i)}$ and $\mathrm{UP}_{R,\mathrm{TRIX}}^{(i)}$ always have the same parameter, it will end up with the same $\boldsymbol{Z}_{h,r}^{(k_1)}$.

For the last k_2 layers, InGram only updates entity embedding. InGram's message passing function on the original knowledge graph is:

$$\begin{aligned}
\boldsymbol{X}_{h,r}^{(i)}(u) &= \mathrm{UP}_{V,\mathrm{InGram}}^{(i)}\Big(\boldsymbol{X}_{h,r}^{(i-1)}(u), \\
&\qquad \mathrm{AGG}_{V,\mathrm{InGram}}^{(i)}\Big(\mathrm{MSG}_{V,\mathrm{InGram}}^{(i)}(\boldsymbol{X}_{h,r}^{(i-1)}(v), \boldsymbol{Z}_{h,r}^{(k_1)}(r'))|(u,r',v) \in \boldsymbol{A}_V^{\mathrm{InGram}}\Big)\Big)
\end{aligned}$$

For TRIX, we choose the message passing functions on relation graph as follows and change the entity embedding to the same initial entity embedding as InGram:

$$\begin{aligned}
\boldsymbol{X}_{h,r}^{(i+k_1)}(u) &= \mathrm{UP}_{V,\mathrm{TRIX}}^{(i+k_1)}\Big(\boldsymbol{X}_{h,r}^{(i+k_1-1)}(u), \\
&\qquad \mathrm{AGG}_{V,\mathrm{TRIX}}^{(i+k_1)}\Big(\mathrm{MSG}_{V,\mathrm{TRIX}}^{(i+k_1)}(\boldsymbol{X}_{h,r}^{(i+k_1-1)}(v), \boldsymbol{Z}_{h,r}^{(i+k_1-1)}(r'))|(u,r',v) \in \boldsymbol{A}_V^{\mathrm{TRIX}}\Big)\Big) \\
\boldsymbol{Z}_{h,r}^{(i+k_1)}(r') &= \mathrm{UP}_{R,\mathrm{TRIX}}^{(i+k_1)}\Big(\boldsymbol{Z}_{h,r}^{(i+k_1-1)}(r'), \\
&\qquad \mathrm{AGG}_{R,\mathrm{TRIX}}^{(i+k_1)}\Big(\mathrm{MSG}_{R,\mathrm{TRIX}}^{(i+k_1)}(\boldsymbol{Z}_{h,r}^{(i+k_1-1)}(r''), \boldsymbol{X}_{h,r}^{(i+k_1)}(u))|(r',u,r'') \in \boldsymbol{A}_R^{\mathrm{TRIX}}\Big)\Big) \\
&\qquad = \boldsymbol{Z}_{h,r}^{(i+k_1-1)}(r') = \boldsymbol{Z}_{h,r}^{(k_1)}
\end{aligned}$$

If $\mathrm{UP}_{V,\mathrm{InGram}}^{(i)}$ and $\mathrm{UP}_{V,\mathrm{TRIX}}^{(i+k_1)}$ always have the same parameter, the relation and entity embedding from InGram and TRIX are exactly the same. Since the embeddings are the same, any non-isomorphic triplet that can be distinguished by InGram can also be distinguished by TRIX. □

Lemma 2 (TRIX can distinguish triplets ULTRA and InGram cannot)**.** *There exist non-isomorphic triplets that can be distinguished by TRIX but that cannot be distinguished by ULTRA and InGram.*

Proof. **TRIX is more expressive than ULTRA.** Consider the two triplets (v_{21}, r_3, v_{24}) and (v_{21}, r_3, v_{25}) in Figure 6, which are non-isomorphic because $r_1 \neq r_2$, and assume they are not observable (i.e., they do not exist in the input graph). Assume, however, we want to predict the existence of these two, with the first one having label 1 (exists) and the second one having label 0 (does not exist). In practice, this means that the first one represents a missing link in an incomplete KG, while the other does not exist and should not be predicted as missing. The proof shows the impossibility of ULTRA to distinguish these two and therefore to make different predictions for the two.

In the relation graph of ULTRA, r_1 and r_2 are automorphic. This implies that ULTRA will give r_1 and r_2 the same relation embedding. Since this embedding is then used by ULTRA to obtain the entity embeddings, ULTRA will assign v_{24} and v_{25} the same entity embedding. Therefore, ULTRA will assign (v_{21}, r_3, v_{24}) and (v_{21}, r_3, v_{25}) the same representation, and therefore it will necessarily make the same prediction for these two (predict that they both exist or that they both do not exist, even though the ground truth tell us only one exist).

On the contrary, in the relation graph of TRIX, r_1 and r_2 are not automorphic. This implies that TRIX can give r_1 and r_2 different relation embeddings, and consequently v_{24} and v_{25} can get different entity embeddings. Therefore, TRIX can assign (v_{21}, r_3, v_{24}) and (v_{21}, r_3, v_{25}) different representations, and therefore it can make different predictions for these two (predict that only the first exists, as the ground truth). Therefore, we have just shown that there exist two non-isomorphic triplets that TRIX can distinguish but ULTRA cannot.

TRIX is more expressive than InGram. Consider the two triplets (v_7, r_3, v_{10}) and (v_7, r_3, v_{11}) in Figure 7, which are non-isomorphic because $r_1 \neq r_2$, and assume they are not observable (i.e., they do not exist in the input graph). Assume, however, we want to predict the existence of these two, with the first one having label 1 (exists) and the second one having label 0 (does not exist). In practice, this means that the first one represents a missing link in an incomplete KG, while the other does not exist and should not be predicted as missing. The proof shows the impossibility of InGram to distinguish these two and therefore to make different predictions for the two.

In the relation graph of InGram, r_1 and r_2 are automorphic. This implies that InGram will give r_1 and r_2 the same relation embedding. Since this embedding is then used by InGram to obtain the entity embeddings, InGram will assign v_{10} and v_{11} the same entity embedding. Therefore, InGram will assign (v_7, r_3, v_{10}) and (v_7, r_3, v_{11}) the same representation, and therefore it will necessarily make the same prediction for these two (predict that they both exist or that they both do not exist, even though the ground truth tell us only one exist).

On the contrary, in the relation graph of TRIX, r_1 and r_2 are not automorphic. This implies that TRIX can give r_1 and r_2 different relation embeddings, and consequently v_{10} and v_{11} can get different entity embeddings. Therefore, TRIX can assign (v_7, r_3, v_{10}) and (v_7, r_3, v_{11}) different representations, and therefore it can make different predictions for these two (predict that only the first exists, as the

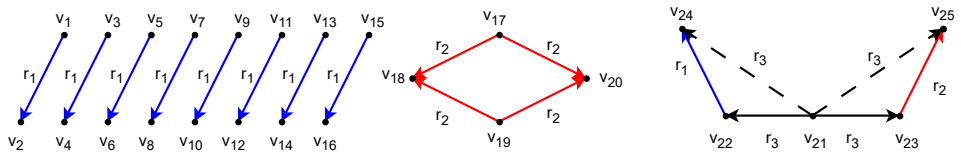


Figure 6: A counter example graph with 10 disconnected components where there exist non-isomorphic triplets that can be distinguished by TRIX but that cannot be distinguished by ULTRA. In the figure, solid lines are edges observed in the knowledge graph and dashed lines are edges we want to predict whether existing or not. As discussed in Appendix B, ULTRA always gives two dashed edges the same embedding while TRIX can distinguish them.

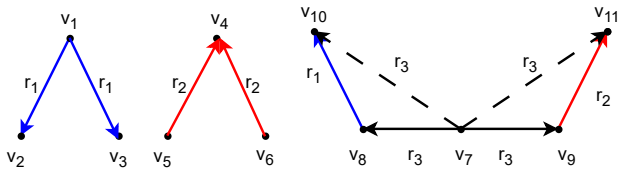


Figure 7: A counter example graph with 3 disconnected components where there exist non-isomorphic triplets that can be distinguished by TRIX but that cannot be distinguished by InGram. In the figure, solid lines are edges observed in the knowledge graph and dashed lines are edges we want to predict whether existing or not. As discussed in Appendix B, InGram always gives two dashed edges the same embedding while TRIX can distinguish them.

ground truth). Therefore, we have just shown that there exist two non-isomorphic triplets that TRIX can distinguish but InGram cannot.

□

Theorem 1 (TRIX is more expressive than ULTRA and InGram)**.** *TRIX is strictly more expressive than ULTRA and InGram in distinguishing between non-isomorphic triplets in KGs.*

Lemma 1 shows that any non-isomorphic triplet that can be distinguished by ULTRA or InGram can also be distinguished by TRIX. Lemma 2 shows that there exist non-isomorphic triplets that can be distinguished by TRIX but that cannot be distinguished by ULTRA or InGram. From Lemmas 1 and 2 directly follows that TRIX is more expressive than ULTRA and InGram.

Proposition 1. *Define* $\alpha = \max_i(\max(\sum_j \mathbb{1}_{\boldsymbol{E}_h[i,j]>0}, \sum_j \mathbb{1}_{\boldsymbol{E}_t[i,j]>0}))$ *which means* α *is the maximum number of unique relations one single entity connects to as the head or the tail. Then the relation adjacency matrix has* $|R|$ *nodes,* $O(|V|\alpha^2)$ *edges and* $|V|$ *relations.*

Proof. The dimension of relation adjacency matrix is $\mathbb{R}^{|R|\times|R|\times|V|\times 4}$. Since each node is affiliated with at most α relations and any two of these relations may generate four non-zero entries (one for head-to-head; one for tail-to-tail; one for head-to-tail; one for tail-to-head) in the relation graph, each node creates at most $4\alpha^2$ non-zero entries in the relation adjacency matrix, implying that the number of nodes is $O(|V|\alpha^2)$. Moreover, the total number of edges in the relation adjacency matrix is at most $4|V|\alpha^2$. □

Now, let us suppose there are L rounds of iterative updates and embedding dimension is d. The time complexity of entity embedding update in TRIX is $O(L|E|d + L|V|d^2)$. The time complexity of relation embedding update in TRIX is $O(L|V|\alpha^2 d + L|R|d^2)$. Since L and d are always $O(1)$, the time complexity of TRIX is $O(|E| + |V|\alpha^2)$ for each query.

The time complexity of relation feature propagation in ULTRA is $O(L|R|^2 d + L|R|d^2)$. The time complexity of entity feature propagation in ULTRA is $O(L|E|d + L|V|d^2)$. Since d and L are always $O(1)$, the time complexity of ULTRA is $O(|E| + |V| + |R|^2)$ for each entity prediction query. However, it goes to $O(|E||R| + |V||R| + |R|^3)$ for each relation prediction query.

Table 4: Statistics of relation graph in pre-training datasets.

	α	# relation	# edge in original graph	# edge in relation graph
WN18RR	7	11	86835	314481
FB15k237	53	237	272115	2247123
CoDExMedium	19	51	185584	706534

For completeness, we report in the following the statistics of the original graph and the relation graph in three exemplary datasets as shown in Table 4. Empirically, the number of edges in the relation graph is usually 4 to 10 times the number of edges in the original graph. This means for entity prediction query, the complexity of ULTRA is up to 10 times better than TRIX. However, in relation prediction, the complexity of TRIX is up to 20 times better than that of ULTRA since ULTRA needs to perform one forward pass for each relation in the relation set.

C Computational Resources

We implemented TRIX using PyTorch [44] (offered under BSD-3 Clause license) and the PyTorch Geometric library [45] (offered under MIT license) for efficient processing of graph-structured data. All experiments were conducted on NVIDIA RTX A5000, NVIDIA RTX A6000, and NVIDIA GeForce RTX 4090 GPUs, and on the Google's Gemini API. For hyperparameter tuning and model selection, we used the Weights and Biases (wandb) library [46].

D LLM Experiment Details

D.1 Details for Task 1

Relation Prediction Prompt of Task 1. In the following task, you will be given background knowledge in the form of triplet (h, r, t) which means entity 'h' has relation 'r' with entity 't'. Then you will be asked some questions about the relationship between entities. Background knowledge: (Kris Kristofferson, occupation, guitarist); (Willow Smith, genre, indie pop);... What is the relationship between entity 'Gaspard Monge' and entity 'France'? Please choose one best answer from the following relations:|parent organization|studies|cause of death|architectural style|unmarried partner|industry|...|. You just need to give the relation and please do not give an explanation.

Entity Prediction Prompt of Task 1. In the following task, you will be given background knowledge in the form of triplet (h, r, t) which means entity 'h' has relation 'r' with entity 't'. Then you will be asked some questions about the relationship between entities. Background knowledge: (Kris Kristofferson, occupation, guitarist); (Willow Smith, genre, indie pop);... Predict the tail entity for triplet (Gaspard Monge, country of citizenship, ?). Please give the 10 most possible answers. You just need to give the names of the entities separated by commas and please do not give explanation.

In the prompt for relation prediction for all the three tasks, all relations in the dataset are listed with '|' as the delimiter. For the sake of simplicity in the presentation, in the next subsections we use "..." to represent the rest of triplets and relations in the prompts.

Table 5 shows the Hits@1 of Gemini-1.5-pro on in-domain relation prediction task. Table 6 shows the Hits@10 of Gemini-1.5-flash on in-domain entity prediction task. We use Gemini-1.5-flash for entity prediction tasks for the sake of reducing costs of experiments. We run the experiment 3 times with the same prompt in English to see if it can generate consistent answers. Gemini performs the task quite well and shows consistency across 3 runs. This indicates given background knowledge in the prompt, LLMs has the capacity to handle in-domain relation and entity predictions well.

D.2 Details for Task 2

Relation Prediction Prompt of Task 2. In the following task, you will first be given background knowledge in the form of triplet (h, r, t) which means entity 'h' has relation 'r' with entity 't'. Then you will be asked some questions about the relationship between entities. Please notice that some words are replaced with metasyntactic words in the following paragraph. Background

Table 5: Task 1: In-domain LLM relation predictions Hits@1 on CoDEx-S.

	Run #1	Run #2	Run #3	Worst
Gemini-1.5-pro	0.933	0.933	0.933	0.933
ULTRA	0.820	0.820	0.820	0.820
TRIX	**0.935**	**0.935**	**0.935**	**0.935**

Table 6: Task 1: In-domain LLM entity predictions Hits@10 on CoDEx-S.

	Run #1	Run #2	Run #3	Worst
Gemini-1.5-flash	0.308	0.308	0.308	0.308
ULTRA	0.667	0.667	0.667	0.667
TRIX	**0.670**	**0.670**	**0.670**	**0.670**

knowledge: (foo, baz, guitarist); (Willow Smith, genre, bar);. . . What is the relationship between entity 'foo' and entity 'bar'? Please choose one best answer from the following relation IDs:|parent organization|studies|quux|baz|. . . |. You just need to give the relation and please do not give an explanation.

Entity Prediction Prompt of Task 2. In the following task, you will first be given background knowledge in the form of triplet (h, r, t) which means entity 'h' has relation 'r' with entity 't'. Then you will be asked some questions about the relationship between entities. Please notice that some words are replaced with metasyntactic words in the following paragraph. Background knowledge: (foo, baz, guitarist); (Willow Smith, genre, bar);. . . Predict the tail entity for triplet (foo, garply, ?). Please give the 10 most possible answers. You just need to give the names of the entities separated by commas and please do not give explanation.

Table 7 shows the Hits@1 of Gemini-1.5-pro on out-domain relation prediction task. Table 8 shows the Hits@10 of Gemini-1.5-flash on out-domain entity prediction task. The neighbor entities of the head entity and the relations that connect the head entity with its neighbors are replaced with metasyntactic words. We run the experiment 3 times with different metasyntactic words but the underlying structural pattern is exactly the same as in the in-domain task. The results indicate the LLM can not do the out-of-domain task well. This means the LLM relied more on the known semantic description of the words instead of the structural pattern of the graph so that when there are new entities and relations, it can not perform inductive reasoning on them.

D.3 Details for Task 3

Relation Prediction Prompt of Task 3. In the following task, entities and relations will be expressed with their IDs. You will first be given the mapping from entities to their IDs and the mapping from relations to their IDs. Then you will be given background knowledge in the form of triplet (h, r, t) which means entity 'h' has relation 'r' with entity 't'. Finally you will be asked some questions about the relationship between entities. Entity mapping: Mireille Darc is entity '8831'; Breton is entity '20512'; Tomas Tranströmer is entity '1641'. . . Relation mapping: located in the administrative terroritorial entity is relation '15' . . . Background knowledge: (15443, 16, 1093); (21198, 16, 9387); (14854, 8, 10218). . . What is the relationship between entity '18127' and entity '1799'? Please choose one best answer from the following relation IDs:|45|48|27|35|. . . |. You just need to give the ID of that relation and please do not give an explanation.

Entity Prediction Prompt of Task 3. In the following task, entities and relations will be expressed with their IDs. You will first be given the mapping from entities to their IDs and the mapping from relations to their IDs. Then you will be given background knowledge in the form of triplet (h, r, t) which means entity 'h' has relation 'r' with entity 't'. Finally you will be asked some questions about the relationship between entities. Entity mapping: Mireille Darc is entity '8831'; Breton is entity '20512'; Tomas Tranströmer is entity '1641'. . . Relation mapping: located in the administrative terroritorial entity is relation '15' . . . Background knowledge: (15443, 16, 1093); (21198, 16, 9387); (14854, 8, 10218). . . Predict the tail entity for triplet (18127, 45, ?). Please give the 10 most possible answers. You just need to give the IDs of the entities separated by commas and please do not give explanation.

Table 7: Task 2: Out-of-domain LLM relation predictions Hits@1 on CoDEx-S. Effects of Metasyntactic Words on Relation Predictions on CoDEx-S.

	Run #1	Run #2	Run #3	Worst
Gemini-1.5-pro	0.667	0.667	0.633	0.633
ULTRA	0.820	0.820	0.820	0.820
TRIX	**0.935**	**0.935**	**0.935**	**0.935**

Table 8: Task 2: Out-of-domain LLM entity predictions Hits@10 on CoDEx-S. Effects of Metasyntactic Words on Entity Predictions on CoDEx-S.

	Run #1	Run #2	Run #3	Worst
Gemini-1.5-flash	0.212	0.250	0.327	0.212
ULTRA	0.667	0.667	0.667	0.667
TRIX	**0.670**	**0.670**	**0.670**	**0.670**

Table 9 shows the Hits@1 of Gemini-1.5-pro on relation prediction in Task 3. Table 10 shows the Hits@10 of Gemini-1.5-flash on entity prediction in Task 3. The entities and relations are expressed as IDs. We run the experiment 3 times with permutated IDs but the underlying structural pattern is exactly the same as in the in-domain task. The results demonstrate that the LLM is very sensitive to ID permutation so that its performance is inconsistent across 3 permutations.

E Datasets

The statistics of all 57 datasets used in the experiments in presented in Tables 11,12,13. All datasets are publicly available under open licenses (MIT or CC-BY).

F Detailed Experiment Results of Entity and Relation Prediction

F.1 Loss Function

TRIX is trained by minimizing the binary cross entropy loss over positive and negative triplets.

For entity prediction, the loss function is:

$$\text{Loss} = -\log p(h, r, t) - \sum_{i=1}^{n} \frac{1}{n} \log(1 - p(h'_i, r, t'_i))$$

where (h, r, t) is the positive triplet and (h'_i, r, t'_i) is a negative triplet by corrupting the head or the tail. n is the number of negative triplets per positive triplet.

For relation prediction, the loss function is:

$$\text{Loss} = -\log p(h, r, t) - \sum_{i=1}^{n} \frac{1}{n} \log(1 - p(h, r'_i, t))$$

where (h, r, t) is the positive triplet and (h, r'_i, t) is a negative triplet by corrupting the relation. n is the number of negative triplets per positive triplet.

F.2 Ablation Study

We conducted multiple experiments to gain deeper insights into the pre-training quality of TRIX and to quantify the impact of the proposed adjacency matrix and the iterative updating scheme on its performance. We compare the performance of (1) TRIX, (2) TRIX without iterative updates, and (3) TRIX without iterative updates and without our relation graph (using the relation graph of ULTRA). We can not do test TRIX w/o proposed relation graph and w/ iterative updates because iterative updates are impossible with the prior relation graphs since entities are not included as edges connecting relations in prior relation graphs and thus the message passing on prior relation graphs

Table 9: Task 3: Out-of-domain LLM relation predictions Hits@1 on CoDEx-S. Effects of Input Permutations on Relation Predictions on CoDEx-S.

	Permutation #1	Permutation #2	Permutation #3	Worst
Gemini-1.5-pro	0.346	0.731	0.615	0.346
ULTRA	0.820	0.820	0.820	0.820
TRIX	**0.935**	**0.935**	**0.935**	**0.935**

Table 10: Task 3: Out-of-domain LLM entity predictions Hits@10 on CoDEx-S. Effects of Input Permutations on Entity Predictions on CoDEx-S.

	Permutation #1	Permutation #2	Permutation #3	Worst
Gemini-1.5-flash	0.212	0.250	0.231	0.212
ULTRA	0.667	0.667	0.667	0.667
TRIX	**0.670**	**0.670**	**0.670**	**0.670**

Table 11: Transductive datasets (16) used in the experiments. Train, Valid, Test denote triples in the respective set. Task denotes the prediction task: *h/t* is predicting both heads and tails, *tails* is only predicting tails.

Dataset	Reference	Entities	Rels	Train	Valid	Test	Task
CoDEx Small	[35]	2034	42	32888	1827	1828	h/t
WDsinger	[47]	10282	135	16142	2163	2203	h/t
FB15k237_10	[47]	11512	237	27211	15624	18150	tails
FB15k237_20	[47]	13166	237	54423	16963	19776	tails
FB15k237_50	[47]	14149	237	136057	17449	20324	tails
FB15k237	[48]	14541	237	272115	17535	20466	h/t
CoDEx Medium	[35]	17050	51	185584	10310	10311	h/t
NELL23k	[47]	22925	200	25445	4961	4952	h/t
WN18RR	[49]	40943	11	86835	3034	3134	h/t
AristoV4	[50]	44949	1605	242567	20000	20000	h/t
Hetionet	[51]	45158	24	2025177	112510	112510	h/t
NELL995	[52]	74536	200	149678	543	2818	h/t
CoDEx Large	[35]	77951	69	551193	30622	30622	h/t
ConceptNet100k	[53]	78334	34	100000	1200	1200	h/t
DBpedia100k	[54]	99604	470	597572	50000	50000	h/t
YAGO310	[55]	123182	37	1079040	5000	5000	h/t

would not use the entity embeddings. The data presented in Table 14 clearly demonstrates that both the proposed relation graph and the iterative update scheme significantly enhance the prediction performance.

F.3 Detailed Results

Table 15 shows an an overview of TRIX performance improvement compared with ULTRA. Table 16 shows detailed zero-shot entity prediction MRR and Hits@10. Table 17 shows detailed fine-tuned entity prediction MRR and Hits@10. Table 18 shows detailed zero-shot relation prediction MRR and Hits@1. Table 19 shows detailed fine-tuned relation prediction MRR and Hits@1. From these tables we can conclude TRIX outperforms the baseline model in both entity and relation prediction.

Table 12: Inductive entity (e) datasets (18) used in the experiments. Triples denote the number of edges of the graph given at training, validation, or test. Valid and Test denote triples to be predicted in the validation and test sets in the respective validation and test graph.

Dataset	Rels	Training Graph		Validation Graph			Test Graph		
		Entities	Triples	Entities	Triples	Valid	Entities	Triples	Test
FB v1 [8]	180	1594	4245	1594	4245	489	1093	1993	411
FB v2 [8]	200	2608	9739	2608	9739	1166	1660	4145	947
FB v3 [8]	215	3668	17986	3668	17986	2194	2501	7406	1731
FB v4 [8]	219	4707	27203	4707	27203	3352	3051	11714	2840
WN v1 [8]	9	2746	5410	2746	5410	630	922	1618	373
WN v2 [8]	10	6954	15262	6954	15262	1838	2757	4011	852
WN v3 [8]	11	12078	25901	12078	25901	3097	5084	6327	1143
WN v4 [8]	9	3861	7940	3861	7940	934	7084	12334	2823
NELL v1 [8]	14	3103	4687	3103	4687	414	225	833	201
NELL v2 [8]	88	2564	8219	2564	8219	922	2086	4586	935
NELL v3 [8]	142	4647	16393	4647	16393	1851	3566	8048	1620
NELL v4 [8]	76	2092	7546	2092	7546	876	2795	7073	1447
ILPC Small [56]	48	10230	78616	6653	20960	2908	6653	20960	2902
ILPC Large [56]	65	46626	202446	29246	77044	10179	29246	77044	10184
HM 1k [57]	11	36237	93364	36311	93364	1771	9899	18638	476
HM 3k [57]	11	32118	71097	32250	71097	1201	19218	38285	1349
HM 5k [57]	11	28601	57601	28744	57601	900	23792	48425	2124
IndigoBM [24]	229	12721	121601	12797	121601	14121	14775	250195	14904

Table 13: Inductive entity and relation (e, r) datasets (23) used in the experiments. Triples denote the number of edges of the graph given at training, validation, or test. Valid and Test denote triples to be predicted in the validation and test sets in the respective validation and test graph.

Dataset	Training Graph			Validation Graph				Test Graph			
	Entities	Rels	Triples	Entities	Rels	Triples	Valid	Entities	Rels	Triples	Test
FB-25 [3]	5190	163	91571	4097	216	17147	5716	4097	216	17147	5716
FB-50 [3]	5190	153	85375	4445	205	11636	3879	4445	205	11636	3879
FB-75 [3]	4659	134	62809	2792	186	9316	3106	2792	186	9316	3106
FB-100 [3]	4659	134	62809	2624	77	6987	2329	2624	77	6987	2329
WK-25 [3]	12659	47	41873	3228	74	3391	1130	3228	74	3391	1131
WK-50 [3]	12022	72	82481	9328	93	9672	3224	9328	93	9672	3225
WK-75 [3]	6853	52	28741	2722	65	3430	1143	2722	65	3430	1144
WK-100 [3]	9784	67	49875	12136	37	13487	4496	12136	37	13487	4496
NL-0 [3]	1814	134	7796	2026	112	2287	763	2026	112	2287	763
NL-25 [3]	4396	106	17578	2146	120	2230	743	2146	120	2230	744
NL-50 [3]	4396	106	17578	2335	119	2576	859	2335	119	2576	859
NL-75 [3]	2607	96	11058	1578	116	1818	606	1578	116	1818	607
NL-100 [3]	1258	55	7832	1709	53	2378	793	1709	53	2378	793
Metafam [58]	1316	28	13821	1316	28	13821	590	656	28	7257	184
FBNELL [58]	4636	100	10275	4636	100	10275	1055	4752	183	10685	597
Wiki MT1 tax [58]	10000	10	17178	10000	10	17178	1908	10000	9	16526	1834
Wiki MT1 health [58]	10000	7	14371	10000	7	14371	1596	10000	7	14110	1566
Wiki MT2 org [58]	10000	10	23233	10000	10	23233	2581	10000	11	21976	2441
Wiki MT2 sci [58]	10000	16	16471	10000	16	16471	1830	10000	16	14852	1650
Wiki MT3 art [58]	10000	45	27262	10000	45	27262	3026	10000	45	28023	3113
Wiki MT3 infra [58]	10000	24	21990	10000	24	21990	2443	10000	27	21646	2405
Wiki MT4 sci [58]	10000	42	12576	10000	42	12576	1397	10000	42	12516	1388
Wiki MT4 health [58]	10000	21	15539	10000	21	15539	1725	10000	20	15337	1703

Table 14: Zero-shot entity prediction results of TRIX, TRIX without proposed relation graph and TRIX without iterative update scheme.

	MRR	Hits@10
TRIX w/o proposed relation graph and w/o iterative updates	0.356	0.508
TRIX w/ proposed relation graph and w/o iterative updates	0.361	0.518
TRIX	**0.390**	**0.548**

Table 15: An overview of TRIX performance improvement compared with ULTRA.

Task	Entity		Relation	
Improvement	zero-shot	finetuned	zero-shot	finetuned
-2% and below	4	3	8	12
-2% to 0%	9	19	2	3
0% to 2%	15	24	6	7
2% and above	26	11	38	35
Total	54	57	54	57

Table 16: Zero-shot entity prediction MRR and Hits@10 over 57 KGs from distinct domains.

Dataset	ULTRA MRR	TRIX MRR	ULTRA Hits@10	TRIX Hits@10
CoDEx Small	**0.472**	**0.472**	0.667	**0.670**
CoDEx Large	**0.338**	0.335	**0.469**	**0.469**
NELL-995	0.406	**0.472**	0.543	**0.629**
YAGO 310	**0.451**	0.409	0.615	**0.627**
WDsinger	0.382	**0.511**	0.498	**0.609**
NELL23k	0.239	**0.290**	0.408	**0.497**
FB15k237_10	**0.248**	0.246	**0.398**	0.393
FB15k237_20	**0.272**	0.269	**0.436**	0.430
FB15k237_50	**0.324**	0.321	**0.526**	0.521
DBpedia100k	0.398	**0.426**	0.576	**0.603**
AristoV4	**0.182**	0.181	0.282	**0.286**
ConceptNet100k	0.082	**0.193**	0.162	**0.345**
Hetionet	0.257	**0.279**	0.379	**0.420**
FB-100	**0.449**	0.436	**0.642**	0.635
FB-75	**0.403**	0.401	0.604	**0.611**
FB-50	**0.338**	0.334	0.543	**0.547**
FB-25	0.388	**0.393**	0.640	**0.650**
WK-100	0.164	**0.188**	0.286	**0.299**
WK-75	0.365	**0.368**	**0.537**	0.513
WK-50	**0.166**	**0.166**	**0.324**	0.313
WK-25	**0.316**	0.305	**0.532**	0.496
NL-100	0.471	**0.486**	0.651	**0.676**
NL-75	**0.368**	0.351	**0.547**	0.525
NL-50	**0.407**	0.404	**0.570**	0.548
NL-25	**0.395**	0.377	0.569	**0.589**
NL-0	0.342	**0.385**	0.523	**0.549**
HM 1k	0.059	**0.072**	0.092	**0.128**
HM 3k	0.037	**0.069**	0.077	**0.119**
HM 5k	0.034	**0.062**	0.071	**0.110**
HM Indigo	**0.440**	0.436	**0.648**	0.645
MT1 tax	0.224	**0.358**	0.305	**0.452**
MT1 health	0.298	**0.376**	0.374	**0.457**
MT2 org	**0.095**	0.091	**0.159**	0.156
MT2 sci	0.258	**0.323**	0.354	**0.465**
MT3 art	0.259	**0.284**	0.402	**0.441**
MT3 infra	0.619	**0.655**	0.755	**0.797**
MT4 sci	0.274	**0.290**	0.449	**0.460**
MT4 health	0.624	**0.677**	0.737	**0.775**
Metafam	0.238	**0.341**	0.644	**0.815**
FBNELL	**0.485**	0.473	0.652	**0.660**
WN-v1	0.648	**0.699**	0.768	**0.791**
WN-v2	0.663	**0.678**	0.765	**0.781**
WN-v3	0.376	**0.418**	0.476	**0.541**
WN-v4	0.611	**0.648**	0.705	**0.723**
FB-v1	0.498	**0.515**	0.656	**0.682**
FB-v2	0.512	**0.525**	0.700	**0.730**
FB-v3	0.491	**0.501**	0.654	**0.669**
FB-v4	0.486	**0.493**	0.677	**0.687**
NL-v1	0.785	**0.806**	**0.913**	0.898
NL-v2	0.526	**0.569**	0.707	**0.768**
NL-v3	0.515	**0.558**	0.702	**0.743**
NL-v4	0.479	**0.538**	0.712	**0.765**
ILPC Small	0.302	**0.303**	0.443	**0.455**
ILPC Large	0.290	**0.307**	0.424	**0.428**

Table 17: Finetuned entity prediction MRR and Hits@10 over 57 KGs from distinct domains.

Dataset	ULTRA MRR	TRIX MRR	ULTRA Hits@10	TRIX Hits@10
WN18RR	0.480 ± 0.000	**0.514** ± 0.003	**0.614** ± 0.000	0.611 ± 0.005
FB15K237	**0.368** ± 0.000	0.366 ± 0.002	**0.564** ± 0.000	0.559 ± 0.002
CoDEx Medium	**0.372** ± 0.000	0.365 ± 0.001	**0.525** ± 0.000	0.521 ± 0.001
CoDEx Small	**0.490** ± 0.003	0.484 ± 0.001	**0.686** ± 0.003	0.676 ± 0.003
CoDEx Large	0.343 ± 0.002	**0.348** ± 0.002	0.478 ± 0.002	**0.481** ± 0.002
NELL-995	**0.509** ± 0.013	0.506 ± 0.030	**0.660** ± 0.006	0.648 ± 0.016
YAGO 310	**0.557** ± 0.009	0.541 ± 0.050	**0.710** ± 0.003	0.702 ± 0.021
WDsinger	0.417 ± 0.002	**0.502** ± 0.001	0.526 ± 0.002	**0.620** ± 0.001
NELL23k	0.268 ± 0.001	**0.306** ± 0.010	0.450 ± 0.001	**0.536** ± 0.007
FB15k237_10	**0.254** ± 0.001	0.253 ± 0.001	**0.411** ± 0.001	0.408 ± 0.002
FB15k237_20	**0.274** ± 0.001	0.273 ± 0.001	**0.445** ± 0.002	0.441 ± 0.001
FB15k237_50	**0.325** ± 0.002	0.322 ± 0.001	**0.528** ± 0.002	0.522 ± 0.002
DBpedia100k	0.436 ± 0.008	**0.457** ± 0.026	0.603 ± 0.006	**0.619** ± 0.012
AristoV4	0.343 ± 0.006	**0.345** ± 0.009	0.496 ± 0.004	**0.499** ± 0.010
ConceptNet100k	0.310 ± 0.004	**0.340** ± 0.008	0.529 ± 0.007	**0.564** ± 0.001
Hetionet	**0.399** ± 0.005	0.394 ± 0.004	**0.538** ± 0.004	0.534 ± 0.005
WN-v1	0.685 ± 0.003	**0.705** ± 0.007	0.793 ± 0.003	**0.798** ± 0.005
WN-v2	0.679 ± 0.002	**0.682** ± 0.004	0.779 ± 0.003	**0.780** ± 0.002
WN-v3	0.411 ± 0.008	**0.425** ± 0.010	**0.546** ± 0.006	0.543 ± 0.006
WN-v4	0.614 ± 0.003	**0.650** ± 0.002	0.720 ± 0.001	**0.722** ± 0.002
FB-v1	0.509 ± 0.002	**0.515** ± 0.000	0.670 ± 0.004	**0.682** ± 0.000
FB-v2	0.524 ± 0.003	**0.525** ± 0.000	0.710 ± 0.004	**0.730** ± 0.000
FB-v3	**0.504** ± 0.001	0.501 ± 0.000	0.663 ± 0.003	**0.669** ± 0.000
FB-v4	**0.496** ± 0.001	0.493 ± 0.000	0.684 ± 0.001	**0.687**± 0.000
NL-v1	0.757 ± 0.021	**0.804** ± 0.007	0.878 ± 0.035	**0.899** ± 0.001
NL-v2	**0.575** ± 0.004	0.571 ± 0.003	0.761 ± 0.007	**0.764** ± 0.006
NL-v3	0.563 ± 0.004	**0.571** ± 0.007	0.755 ± 0.006	**0.759** ± 0.006
NL-v4	0.469 ± 0.020	**0.551** ± 0.001	0.733 ± 0.011	**0.772** ± 0.004
ILPC Small	**0.303** ± 0.001	**0.303** ± 0.001	0.453 ± 0.002	**0.455** ± 0.001
ILPC Large	0.308 ± 0.002	**0.310** ± 0.002	**0.431** ± 0.001	**0.431** ± 0.003
HM 1k	0.042 ± 0.002	**0.072** ± 0.000	0.100 ± 0.007	**0.128** ± 0.000
HM 3k	0.030 ± 0.002	**0.069** ± 0.000	0.090 ± 0.003	**0.119** ± 0.000
HM 5k	0.025 ± 0.001	**0.074** ± 0.021	0.068 ± 0.003	**0.118** ± 0.013
HM Indigo	0.432 ± 0.001	**0.436** ± 0.000	0.639 ± 0.002	**0.645** ± 0.000
FB-100	**0.444** ± 0.003	0.436 ± 0.001	**0.643** ± 0.004	0.633 ± 0.003
FB-75	0.400 ± 0.003	**0.401** ± 0.000	0.598 ± 0.004	**0.611** ± 0.000
FB-50	**0.334** ± 0.002	**0.334** ± 0.000	0.538 ± 0.004	**0.547** ± 0.000
FB-25	0.383 ± 0.001	**0.393** ± 0.000	0.635 ± 0.002	**0.650** ± 0.000
WK-100	0.168 ± 0.005	**0.188** ± 0.000	0.286 ± 0.003	**0.299** ± 0.000
WK-75	**0.380** ± 0.001	0.368 ± 0.000	**0.530** ± 0.009	0.513 ± 0.000
WK-50	0.140 ± 0.010	**0.166** ± 0.000	0.280 ± 0.012	**0.313** ± 0.000
WK-25	**0.321** ± 0.003	0.300 ± 0.009	**0.535** ± 0.007	0.493 ± 0.006
NL-100	0.458 ± 0.012	**0.482** ± 0.002	0.684 ± 0.011	**0.691** ± 0.001
NL-75	**0.374** ± 0.007	0.351 ± 0.000	**0.570** ± 0.005	0.525 ± 0.000
NL-50	**0.418** ± 0.005	0.405 ± 0.002	**0.595** ± 0.005	0.555 ± 0.012
NL-25	**0.407** ± 0.009	0.377 ± 0.000	**0.596** ± 0.012	0.589 ± 0.000
NL-0	0.329 ± 0.010	**0.385** ± 0.000	**0.551** ± 0.012	0.549 ± 0.000
MT1 tax	0.330 ± 0.046	**0.397** ± 0.001	0.459 ± 0.056	**0.508** ± 0.002
MT1 health	**0.380** ± 0.002	0.376 ± 0.000	**0.467** ± 0.006	0.457 ± 0.000
MT2 org	**0.104** ± 0.001	0.098 ± 0.002	**0.170** ± 0.001	0.162 ± 0.002
MT2 sci	0.311 ± 0.010	**0.331** ± 0.012	0.451 ± 0.042	**0.526** ± 0.005
MT3 art	**0.306** ± 0.003	0.289 ± 0.004	**0.473** ± 0.003	0.441 ± 0.001
MT3 infra	0.657 ± 0.008	**0.672** ± 0.003	0.807 ± 0.007	**0.810** ± 0.002
MT4 sci	0.303 ± 0.007	**0.305** ± 0.003	0.478 ± 0.003	**0.482** ± 0.001
MT4 health	**0.704** ± 0.002	0.702 ± 0.002	**0.785** ± 0.002	**0.785** ± 0.002
Metafam	**0.997** ± 0.003	**0.997** ± 0.003	**1.000** ± 0.000	**1.000** ± 0.000
FBNELL	**0.481** ± 0.004	0.478 ± 0.004	**0.661** ± 0.011	0.655 ± 0.012

Table 18: Zero-shot relation prediction MRR and Hits@1 over 57 KGs from distinct domains.

Dataset	ULTRA MRR	TRIX MRR	ULTRA Hits@1	TRIX Hits@1
CoDEx Small	0.900	**0.961**	0.820	**0.935**
CoDEx Large	0.892	**0.902**	0.824	**0.837**
NELL-995	**0.583**	0.578	0.437	**0.457**
YAGO 310	0.646	**0.783**	0.403	**0.598**
WDsinger	0.668	**0.720**	0.546	**0.621**
NELL23k	0.669	**0.756**	0.548	**0.657**
FB15k237_10	0.688	**0.795**	0.550	**0.711**
FB15k237_20	0.695	**0.834**	0.558	**0.758**
FB15k237_50	0.717	**0.876**	0.591	**0.812**
DBpedia100k	0.650	**0.717**	0.509	**0.582**
AristoV4	0.254	**0.389**	0.201	**0.265**
ConceptNet100k	0.181	**0.650**	0.083	**0.469**
Hetionet	0.634	**0.809**	0.524	**0.707**
WN-v1	**0.836**	0.792	**0.740**	0.613
WN-v2	**0.853**	0.764	**0.790**	0.572
WN-v3	0.707	**0.741**	**0.577**	0.568
WN-v4	**0.860**	0.764	**0.803**	0.570
FB-v1	0.646	**0.705**	0.523	**0.599**
FB-v2	0.695	**0.713**	0.570	**0.590**
FB-v3	0.679	**0.742**	0.553	**0.644**
FB-v4	0.638	**0.766**	0.488	**0.665**
NL-v1	0.636	**0.657**	0.358	**0.453**
NL-v2	0.742	**0.780**	0.652	**0.696**
NL-v3	0.669	**0.725**	0.544	**0.612**
NL-v4	0.606	**0.794**	0.489	**0.691**
ILPC Small	0.905	**0.919**	0.843	**0.872**
ILPC Large	0.875	**0.894**	0.799	**0.829**
HM 1k	0.626	**0.663**	**0.447**	0.414
HM 3k	0.592	**0.664**	**0.439**	0.418
HM 5k	0.605	**0.672**	**0.452**	0.428
HM Indigo	0.681	**0.852**	0.559	**0.765**
FB-100	0.830	**0.921**	0.728	**0.880**
FB-75	0.698	**0.822**	0.555	**0.747**
FB-50	0.696	**0.780**	0.575	**0.699**
FB-25	0.687	**0.805**	0.565	**0.724**
WK-100	0.887	**0.907**	0.812	**0.869**
WK-75	0.911	**0.916**	0.875	**0.883**
WK-50	0.865	**0.868**	0.793	**0.818**
WK-25	0.857	**0.881**	0.760	**0.823**
NL-100	0.743	**0.884**	0.564	**0.796**
NL-75	**0.795**	0.788	0.692	**0.699**
NL-50	0.680	**0.755**	0.569	**0.636**
NL-25	0.688	**0.742**	0.562	**0.614**
NL-0	0.632	**0.658**	0.502	**0.519**
MT1 tax	**0.985**	0.975	**0.976**	0.958
MT1 health	0.721	**0.973**	0.561	**0.949**
MT2 org	0.974	**0.986**	0.951	**0.973**
MT2 sci	**0.976**	0.964	**0.961**	0.941
MT3 art	0.881	**0.885**	0.798	**0.825**
MT3 infra	**0.962**	0.940	**0.935**	0.905
MT4 sci	0.933	**0.966**	0.891	**0.944**
MT4 health	0.826	**0.937**	0.719	**0.898**
Metafam	0.124	**0.291**	0.000	**0.011**
FBNELL	0.700	**0.726**	0.564	**0.605**

Table 19: Finetuned relation prediction MRR and Hits@1 over 57 KGs from distinct domains.

Dataset	ULTRA MRR	TRIX MRR	ULTRA Hits@1	TRIX Hits@1
WN18RR	**0.914** ± 0.004	0.783 ± 0.009	**0.871** ± 0.001	0.634 ± 0.007
FB15K237	0.795 ± 0.017	**0.924** ± 0.005	0.709 ± 0.025	**0.870** ± 0.024
CoDEx Medium	0.919 ± 0.032	**0.931** ± 0.001	0.870 ± 0.048	**0.886** ± 0.001
CoDEx Small	0.942 ± 0.007	**0.964** ± 0.002	0.900 ± 0.014	**0.943** ± 0.002
CoDEx Large	0.907 ± 0.000	**0.908** ± 0.003	**0.850** ± 0.000	0.845 ± 0.004
NELL-995	**0.630** ± 0.000	0.578 ± 0.000	**0.513** ± 0.000	0.457 ± 0.000
YAGO 310	**0.930** ± 0.002	0.826 ± 0.000	**0.891** ± 0.004	0.666 ± 0.000
WDsinger	**0.730** ± 0.012	0.721 ± 0.004	0.603 ± 0.020	**0.627** ± 0.007
NELL23k	0.688 ± 0.008	**0.755** ± 0.004	0.571 ± 0.009	**0.658** ± 0.005
FB15k237_10	0.688 ± 0.000	**0.795** ± 0.000	0.550 ± 0.000	**0.711** ± 0.000
FB15k237_20	0.695 ± 0.000	**0.846** ± 0.011	0.558 ± 0.000	**0.778** ± 0.017
FB15k237_50	0.728 ± 0.013	**0.903** ± 0.003	0.618 ± 0.027	**0.858** ± 0.006
DBpedia100k	0.650 ± 0.000	**0.780** ± 0.003	0.509 ± 0.000	**0.665** ± 0.006
AristoV4	0.254 ± 0.000	**0.498** ± 0.002	0.201 ± 0.000	**0.381** ± 0.002
ConceptNet100k	0.612 ± 0.000	**0.712** ± 0.005	0.488 ± 0.000	**0.551** ± 0.003
Hetionet	0.737 ± 0.031	**0.922** ± 0.002	0.646 ± 0.041	**0.862** ± 0.005
WN-v1	**0.844** ± 0.021	0.776 ± 0.021	**0.754** ± 0.029	0.591 ± 0.034
WN-v2	**0.834** ± 0.008	0.765 ± 0.009	**0.766** ± 0.013	0.574 ± 0.015
WN-v3	0.707 ± 0.000	**0.756** ± 0.044	0.577 ± 0.000	**0.594** ± 0.064
WN-v4	**0.861** ± 0.005	0.804 ± 0.013	**0.795** ± 0.007	0.651 ± 0.026
FB-v1	0.650 ± 0.008	**0.705** ± 0.000	0.513 ± 0.014	**0.599** ± 0.000
FB-v2	0.675 ± 0.035	**0.713** ± 0.000	0.547 ± 0.040	**0.590** ± 0.000
FB-v3	0.677 ± 0.007	**0.742** ± 0.000	0.556 ± 0.006	**0.644** ± 0.000
FB-v4	0.690 ± 0.026	**0.766** ± 0.000	0.560 ± 0.035	**0.665** ± 0.000
NL-v1	**0.719** ± 0.061	0.590 ± 0.036	**0.504** ± 0.113	0.341 ± 0.066
NL-v2	0.668 ± 0.064	**0.811** ± 0.000	0.549 ± 0.090	**0.740** ± 0.000
NL-v3	0.646 ± 0.014	**0.757** ± 0.004	0.484 ± 0.022	**0.643** ± 0.009
NL-v4	0.570 ± 0.030	**0.822** ± 0.011	0.412 ± 0.056	**0.735** ± 0.011
ILPC Small	**0.922** ± 0.001	0.919 ± 0.000	**0.876** ± 0.001	0.872 ± 0.000
ILPC Large	0.875 ± 0.000	**0.894** ± 0.000	0.799 ± 0.000	**0.829** ± 0.000
HM 1k	0.626 ± 0.000	**0.663** ± 0.000	**0.447** ± 0.000	0.414 ± 0.000
HM 3k	0.592 ± 0.000	**0.664** ± 0.000	**0.439** ± 0.000	0.418 ± 0.000
HM 5k	0.605 ± 0.000	**0.672** ± 0.000	**0.452** ± 0.000	0.428 ± 0.000
HM Indigo	0.726 ± 0.005	**0.835** ± 0.002	0.614 ± 0.004	**0.746** ± 0.003
FB-100	0.851 ± 0.006	**0.921** ± 0.000	0.769 ± 0.016	**0.880** ± 0.000
FB-75	0.754 ± 0.020	**0.822** ± 0.000	0.638 ± 0.032	**0.747** ± 0.000
FB-50	0.696 ± 0.000	**0.780** ± 0.000	0.575 ± 0.000	**0.699** ± 0.000
FB-25	0.684 ± 0.021	**0.805** ± 0.000	0.563 ± 0.024	**0.724** ± 0.000
WK-100	**0.924** ± 0.003	0.916 ± 0.001	0.879 ± 0.002	**0.885** ± 0.003
WK-75	0.911 ± 0.000	**0.937** ± 0.003	0.875 ± 0.000	**0.910** ± 0.003
WK-50	0.865 ± 0.000	**0.881** ± 0.007	0.793 ± 0.000	**0.840** ± 0.014
WK-25	0.897 ± 0.002	**0.905** ± 0.007	0.834 ± 0.005	**0.860** ± 0.011
NL-100	0.803 ± 0.008	**0.885** ± 0.005	0.678 ± 0.012	**0.793** ± 0.008
NL-75	**0.795** ± 0.000	0.790 ± 0.000	0.678 ± 0.000	**0.671** ± 0.000
NL-50	**0.808** ± 0.000	0.774 ± 0.000	0.704 ± 0.000	**0.683** ± 0.000
NL-25	**0.737** ± 0.000	0.709 ± 0.000	0.622 ± 0.000	**0.606** ± 0.000
NL-0	0.632 ± 0.000	**0.655** ± 0.006	0.502 ± 0.000	**0.518** ± 0.002
MT1 tax	0.990 ± 0.001	**0.995** ± 0.001	0.984 ± 0.001	**0.990** ± 0.001
MT1 health	0.929 ± 0.044	**0.973** ± 0.000	0.867 ± 0.087	**0.949** ± 0.000
MT2 org	0.981 ± 0.014	**0.987** ± 0.001	0.963 ± 0.027	**0.978** ±0.001
MT2 sci	0.977 ± 0.001	**0.990** ± 0.001	0.961 ± 0.001	**0.984** ± 0.002
MT3 art	**0.907** ± 0.012	0.887 ± 0.003	0.851 ± 0.024	**0.828** ± 0.005
MT3 infra	0.966 ± 0.003	**0.970** ± 0.001	0.947 ± 0.006	**0.952** ± 0.003
MT4 sci	0.954 ± 0.002	**0.972** ± 0.001	0.929 ± 0.002	**0.952** ± 0.001
MT4 health	0.951 ± 0.006	**0.986** ± 0.001	0.919 ± 0.010	**0.979** ± 0.002
Metafam	**0.368** ± 0.029	0.265 ± 0.044	**0.112** ± 0.036	0.024 ± 0.022
FBNELL	0.720 ± 0.013	**0.766** ± 0.004	0.576 ± 0.020	**0.639** ± 0.006