MuTATE: Unifying Knowledge Graph Embedding and Multi-Task Learning via Counterfactual Residuals

Anonymous Author(s)

ABSTRACT

In this paper, we propose MuTATE, a Multi-Task Augmented paradigm to learn Transferrable Embeddings of knowledge graphs. Previous research efforts can be categorized into two distinct directions. The first assumes that the knowledge graph is complete and uses it to augment specific machine learning models in a task-specific parameterized manner. The second applies geometric, relation-based, and path-based hypotheses to complete and enrich the knowledge graph by learning informative node embeddings. Contrary to these efforts, we propose a novel framework to connect task-specific inductive models with knowledge graph completion and enrichment processes. Specifically, models that predict links between distinct, potentially disconnected subsets of nodes in the knowledge graph are unified vis-a-vis the underlying node embedding space, permitting multi-directional knowledge transfer (model-to-graph, graph-to-model, and even model-to-model via graph) through counterfactual reasoning. Experimental results on two public datasets show that the proposed approach effectively integrates task-models with knowledge graph embedding and permits for the above types of knowledge transfer.

KEYWORDS

Knowledge Graph; Knowledge Graph Embedding; Multi-Task Learning; Counterfactual Reasoning

1 INTRODUCTION

The modern-day rise of artificial intelligence in academic research and industrial applications has sparked renewed interest in knowledge graphs (KGs). Knowledge graphs are graph-structured knowledge bases where vast amounts of information available in the world are succinctly represented as entities (nodes) and relationships (edges)—see Figure 1. Knowledge graphs are semantically enriched (i.e., entities and relationships have associated meanings) and thus unites machine learning and graph technologies to give artificial intelligence the context it needs. Knowledge graphs are essential resources in many vital applications today. For example, intelligent assistants Apple's Siri and Amazon's Alexa, question answering features of modern search engines Google and Microsoft Bing, product recommendation and discovery features of e-commerce marketplaces Amazon and eBay.

Knowledge graphs can express heterogeneous knowledge in various domains in a usable form and satisfy many use cases for domains ranging from linguistics [34], biomedicine [7] to finance [5]. Figure 1 is a toy example of a knowledge graph that captures user attributes, e.g., *age-group*, and item/book attributes, e.g., *genre*, in addition to the *like* relationship between them. A natural consequence of such a generalizable representation is that they exhibit the characteristics of the underlying data such as sparsity for some entities or some types of entities, skew in the volume and types

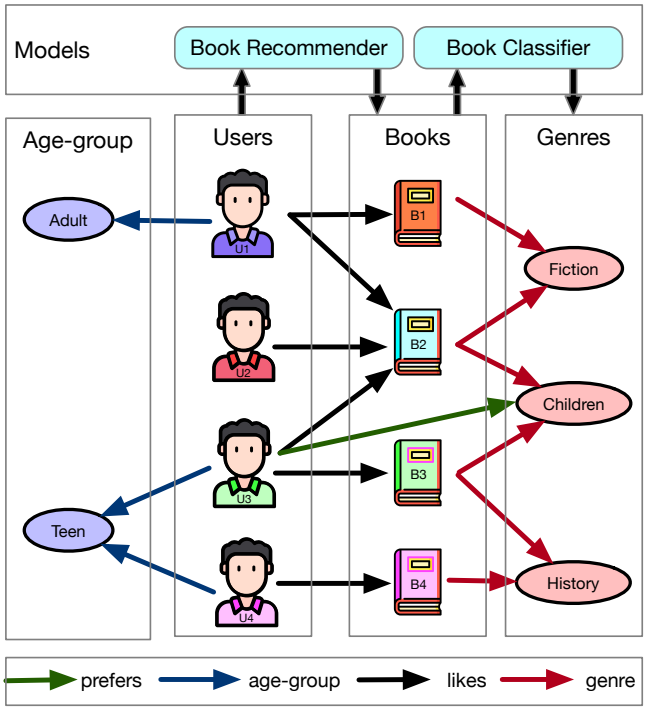

Figure 1: A toy example of a user-item knowledge graph with four types of entities: *user*, *book*, *user age-group*, and *book genre*. The entities are connected via four types of relationships: *prefers*, *in age-group*, *likes*, and *genre*. Models *Book Recommender* and *Book Classifier* recommends books to users and identifies book genres respectively.

of relationships connecting entities. For example, the age-groups relationship of all the people in the knowledge graph is not known. These ubiquitous distributional challenges persist across several domains [17, 20]. Thus, the constructed knowledge graphs are far from complete and mandate continuous enrichment and curation. Note that enriching the knowledge graph can help augment the underlying data too [11]. We categorize the previous research efforts into three distinct directions and describe our innovations.

The first is knowledge graph embedding [2, 30, 35], which attempts to enrich the knowledge graph and incorporate latent structural proximities of nodes by transitively learning a range of simple heuristic patterns among the nodes such as *symmetry*, *anti-symmetry*, *composition* and *analogy* (described in details in Section 4.1). These patterns are, however, unable to distinguish the different relation types and are applied in an equivalent manner to all of them. This can lead to contradictory and incorrect inferences, which may violate the domain knowledge. In our toy example, a user may like a book, and the book may be connected to an author,

but this does not necessarily imply that the user likes the author. If we have a more detailed external task model, which can provide us this feedback, we can avoid making this incorrect inference. Thus, our solution adopts the utility of these patterns, but more importantly, *provides the ability to correct these mistakes across different relation types using external model feedback.*

The second direction views the knowledge graph as an oracle and develops task-specific models that leverage existing connectivity patterns to improve performance for tasks such as question-answering [10] and recommendation [33]. The key challenge in this scenario is that the view of the knowledge graph is not optimized to the task-model's specific architecture. Conversely, the inductive task-models cannot be directly leveraged to densify or improve the knowledge graph either. Our approach addresses this shortcoming as well. We *optimize the view of the knowledge graph so that it is best suited to the specific task-model using counterfactual residual learning*, as described in Section 5.1.

A third recent direction includes some hybrid solutions that bridge the first two directions for specific fixed tasks by simultaneously performing task augmentation and graph enrichment [3, 8]. Such solutions are often predicated on some very specific architectural assumptions about the nature of the task/task-model or external feedback. They do not extend to the broader multi-task setting where the different tasks correspond to the graph nodes' subsets. They hence cannot be bi-directionally integrated with the knowledge graph. We make *no assumptions about the nature or architecture or training objective of the external task-model.* We are thus able to leverage a much more comprehensive range of task-models to improve our knowledge graph.

Our paper proposes a holistic solution to subsume multi-task learning and knowledge graph enrichment via multi-directional knowledge transfer (model-to-graph, graph-to-model, and even model-to-model via graph) with the notion of counterfactual association learning. The key idea is to view each task model as an intervention applied to the knowledge graph entities, much akin to a patient receiving a specific medicine [15, 29]. The counterfactual question then becomes: if we know how the patient reacted to one of the two scenarios (receiving or not receiving the medicine), can we use that outcome alone to accurately predict the counterfactual scenario? This question is directly applicable to link prediction and model enhancement tasks as two complementary aspects of the factual-counterfactual question. When we apply a task-model to an entity, the task-model predicts several connections for the entity.

In Figure 1, when we use *Book Recommender* to recommend books for the user U3, we get B4 as a recommended book. This recommendation enables us to create a new connection between the user U3 and the book B4—see Figure 2. Such connections are counterfactual because the nature of the task-model biases them. Hence, we can formulate it as a causal inference question [19, 24] of whether the suggested links originate purely from the task or task-model eccentricities or vice-versa. The counterfactual question that we propose here is: how can we leverage the task or intervention-biased connections to enrich the underlying knowledge graph, i.e., infer the factual links? Conversely, the opposite direction is: given the factual links of an entity, what are the likely counterfactual links suggested by the model? While the counterfactual to factual link mapping direction enables us to enrich the knowledge graph,

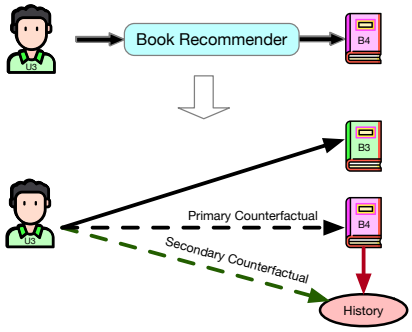

Figure 2: We use *Book Recommender* model to infer counterfactual edges (shown using dotted lines), thereby enriching the KG. Primary counterfactual links are inferred directly from the model. Secondary counterfactual links are infered from one-hop neighbours of the primary links.

the opposite direction lets us improve the task models based on the graph's factual links. In this way, the forward and reverse transformations enable bi-directional knowledge transfer between task models and the knowledge graph.

In this paper, we make the following contributions:

Merging Multi-Task Learning and Knowledge Graph Enrichment/Embedding: This paper proposes a holistic view of knowledge graphs and multi-task learning that permits for multi-directional transfer of knowledge between domain-specific knowledge graphs and task-models. This holistic view overcomes the key limitations of past work. While enabling for bidirectional knowledge transfer, we also do not make assumptions about the nature of the specific task-models, architectures, or objectives.

Generalizability: The proposed framework is highly generalizable; we make no assumptions about the data-domain or the task-models connected to it. As a result, we can integrate very diverse tasks, and their model architectures through a common set of underlying knowledge embeddings. In our experiments, we exhibit this capability with two very distinct models, one recommendation model connecting users and items, and a second item-content model that attempts to predict the most likely words to describe a specific item. We show that we can leverage counterfactual updates obtained from the prediction model to significantly improve the recommendation model's performance for sparse users (in other words, the item-word links are leveraged to form user-item links).

Modeling the Multi-Task Updates to the Graph as a Residual Function: We identify the connection between multi-task knowledge graph updates and covariate domain shift theory [15], which permits us to model different task-specific distributions with the same underlying knowledge graph via residual learning, in a very inexpensive manner.

Strong Experimental Results: We demonstrate strong experimental results with knowledge graphs constructed from two large distinct knowledge graphs, induced on the *Google Local Reviews Dataset* [1][9, 28] and the *Yelp Challenge Dataset* [2]. We show how to

[1]http://cseweb.ucsd.edu/~jmcauley/datasets.html
[2]https://www.yelp.com/dataset/challenge

leverage two very different external task models, word2vec [23] and a context-aware recommender [16] to densify and improve the knowledge graph, and also simultaneously perform model-to-model cross-training (i.e., use the first model to generate feedback that updates the graph, which can then be used to improve the second model). On the whole, we show strong results on graph completion (5% relative to state of the art embedding baselines) and show significant potential for knowledge transfer.

We now summarize related work, formalize our problem, describe our solution, and evaluate the proposed framework.

2 RELATED WORK

Knowledge graphs are essential resources for many AI tasks today. While one branch of research considers the knowledge graph as an oracle and develops machine learning models that leverage existing connectivity patterns to improve task outcome, they often suffer from incompleteness. A variety of representation-based/embedding methods - tensor factorization based and neural network based - have been developed that attempts to enrich the knowledge graph and incorporate latent structural proximities of nodes by transitively learning a range of simple heuristic patterns among the nodes [2, 13, 14, 21, 26, 27, 30, 35]. These patterns are unable to distinguish the different relation types and are applied in an equivalent manner to all of them. Thus, it can lead to contradictory and incorrect inferences, which in turn, may violate the domain knowledge. Additionally, some of these methods are also not suited to handle unbalanced heteerogeneous graphs.

Several recent efforts have attempted to leverage the knowledge graph structure for recommendation [1, 31, 33]. The methods are either path-based that feed the high-order information to the predictive model or regularization-based that leverages the network structure to regularize the recommender model learning. The methods are typically not optimized for the recommendation objective. Conversely, the inductive task-models cannot be directly leveraged to densify or improve theknowledge graph either. Other tasks such as search personalization [25] and question-answering [10] also suffer from similar drawbacks.

In this paper, we propose a holistic solution that addresses the above issues by subsuming multi-task learning and knowledge graph enrichment via multi-directional knowledge transfer via counterfactual association learning.

3 PROBLEM DEFINITION

In this section, we present the distinct components associated with our knowledge graph representations. This representation space establishes a consistent interface between the set of factual links across entities and the task-models that make predictions associating different subsets of the interacting entities in the knowledge graph. However, there are several fundamental differences between the modeling effort for these two types of edges. While we can directly incorporate the factual edges in the node representations, the edges suggested by task-models are counterfactual. In Section 4, we describe our reasoning framework to incorporate the counterfactual edges and infer changes in the node representations.

Knowledge Graph Notations: We consider a heterogeneous directed knowledge graph with multiple types of entities (or sets of nodes) and relations.Let us represent entity types as:

$$\mathbf{E}_1 \text{ (e.g., users)}, \mathbf{E}_2 \text{ (e.g., items)} \cdots \mathbf{E}_{|\mathcal{E}|}$$

where $\mathcal{E} = \{\mathbf{E}_1, \mathbf{E}_2 \cdots \mathbf{E}_{|\mathcal{E}|}\}$ is the set of all entity types. The set of all nodes in the graph is $\cup \mathbf{E}_i$. Let $\mathcal{R} = \{\mathbf{R}_1, \mathbf{R}_2 \cdots \mathbf{R}_{|\mathcal{R}|}\}$ denote the set of relations where each relation $\mathbf{R}_r : \mathbf{E}_1^r \rightarrow \mathbf{E}_2^r$ is a collection of links between two entity types $\mathbf{E}_1^r, \mathbf{E}_2^r \in \mathcal{E}$. Note that, two different relations can exist between the same pair of entity sets.

We denote each edge as (e_1, r, e_2) where $e_1 \in \mathbf{E}_1^r$, $e_2 \in \mathbf{E}_2^r$ denote the head and tail entities respectively and r their relation-type. We denote the respective d-dimensional entity embeddings by adding an overhead arrow to the above notation, i.e., $\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2$. Each relation-type r is described by its head and tail projectors $(\vec{\mathbf{p}}_1^r, \vec{\mathbf{p}}_2^r)$, which are also d-dimensional like the entity embeddings.

Task Model Notations: For simplicity, we only consider discrete prediction models in our analysis. However, regression models can be discretized to fit a similar abstraction. We consider each prediction model $\mathcal{M}^j$ to take an input entity $e_1 \in \mathbf{E}_1^j$ and produce a predicted output entity $e_2 \in \mathbf{E}_2^j$, thus inducing a connection across the two entity sets $\mathbf{E}_1^j, \mathbf{E}_2^j \in \mathcal{E}$ depending on its specific prediction task. For example, a recommendation model could connect the user nodes to the respective recommended item nodes in our knowledge graph. We will refer to these connections as task-biased or intervention-biased counterfactual links since they may not exist in the knowledge graph, but are predicted by the task model. For simplicity, we only consider task-models that establish connections between pairs of entity sets, although the proposed framework is general to multi-variate scenarios.

In the rest of Section 4, we use the subscripts 1 and 2 to denote head and tail entities of a link, while we use the notation r to denote a relation type. Analogously, we use the notation j to denote a task model, $\mathcal{M}^j : \mathbf{E}_1^j \rightarrow \mathbf{E}_2^j$.

4 KNOWLEDGE GRAPH EMBEDDINGS

This section first describes a base embedding model to form a highly scalable and expressive embedding space for multi-task augmentation. Then, we show how to selectively combine the covariant aspects of different prediction models with the same latent embedding space. Finally, we introduce our approach to leverage node embeddings for cross-modal transfer across different tasks.

4.1 Embedding Model

Since knowledge graphs are often incomplete and exhibit a lack of links for a substantial proportion of the nodes, embedding models attempt to infer missing links via proximities in the latent space. Sun et al. [30] describe three fundamental connectivity pattens, *symmetry/antisymmetry*, *composition* and *inversion* which can be stacked by the learned models to encode higher-order patterns:

- **Symmetry:** $(e_1, r_a, e_2) \implies (e_2, r_a, e_1)$
- **Anti-Symmetry:** $(e_1, r_a, e_2) \implies not\,(e_2, r_a, e_1)$
- **Analogy:** (e_1, r_a, e_2) *and* $(e_3, r_a, e_4) \implies (e_1, r_b, e_3) / (e_2, r_c, e_4)$
- **Inversion:** $(e_1, r_a, e_2) \implies (e_2, r_b, e_1)$
- **Composition:** (e_1, r_a, e_2) *and* $(e_2, r_b, e_3) \implies (e_1, r_c, e_3)$

While these patterns enable a good first-cut link selection, they do not distinguish the different relation types and are applied in an

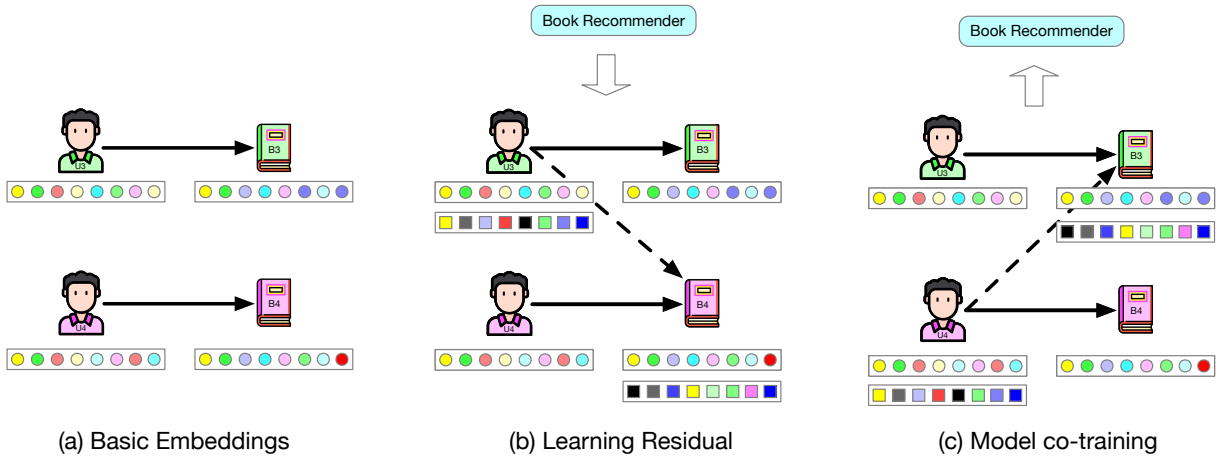

Figure 3: (a) We learn the node embedddings using Equation (3) for fatual links. (b) We use the *Book Recommender* model to infer counterfactual edges and train residuals using Equation (13). (c) We impove the *Book Recommender* model using residuals in Equation (20).

equivalent manner to all of them. For example, the learned patterns may often contradict domain knowledge for some types of links. Prior knowledge graph embedding methods do not provide any mechanism to overcome these challenges in large heterogenous knowledge graph. Our fundamental hypothesis is that leveraging the inductive bias of an external model designed for a specific task can help filter the encoded patterns. Further, in a heterogenous knowledge graph, the degree of sparsity may not be evenly spread across the different node and relation modalities. Thus, cross-modal transfer is particularly important in any enrichment or completion effort, specifically, the following type of cross-modal learning:

- How do we leverage (e_1, r_a, e_2) for predictions of the form (e_1, r', e'), (e_2, r', e'), (e'', r'', e_1), (e'', r'', e_2)?

Note that the answer to the above form of cross-modal learning is specific to the relation types r_a, r', r'' as well the entity nodes and thus can be answered effectively by leveraging external models that are designed and trained for prediction tasks involving either these entities or relations.

In addition to these properties, the sizes of most knowledge graphs can exceed millions of nodes, and billions of edges. Thus efficient parallelization in the embedding model is of utmost importance for most practical applications. DistMult [36] is one of the most successful models in the literature, owing to its simplicity and ability to be block optimized as described by Lerer et al. [18]. However, the DistMult model's key weakness is its inability to model anti-symmetry and composition owing to its formulation, as pointed out in Rotate [30]. On the other hand, Rotate does not treat the head and tail entities uniformly and thus poses scalability constraints with regard to some block optimizations [18]. In our work, we consider both these perspectives in the base embedding model, which is subsequently augmented in the multi-task setting. We essentially apply a simple modification to DistMult to capture both anti-symmetry and composition in the heterogeneous node setting.

The basic DistMult model follows a bilinear function with a learned diagonal projector matrix $(\mathbf{P}_r)$ representing the relation type r. Thus the likelihood of an edge (e_1, r, e_2) is given by:

$$\mathcal{L}(\vec{\mathbf{e}}_1, r, \vec{\mathbf{e}}_2) = \vec{\mathbf{e}}_1^T \mathbf{P}_r \vec{\mathbf{e}}_2 \tag{1}$$

Due to this transformation's symmetric nature, anti-symmetry and inversion are hard to encode in this form. Instead, we apply a simple modification, as explained next. We can change the above function's symmetric nature by describing a head and tail dual-projector form for each relation. Note that this form only involves a few additional parameters, namely twice as many parameters for the relation embeddings. However, in most knowledge graphs, the types of relations are several orders of magnitude smaller than the number of nodes, and thus this parameter overhead is negligible. Thus we define the likelihood of an edge (e_1, r, e_2) as:

$$\mathcal{L}(\vec{\mathbf{e}}_1, r, \vec{\mathbf{e}}_2) = sim((\vec{\mathbf{e}}_1^T \otimes \vec{\mathbf{p}}_1^r)(\vec{\mathbf{e}}_2^T \otimes \vec{\mathbf{p}}_2^r)) \tag{2}$$

It is easy to see that this change enables composition, inversion, and anti-symmetry. The proof is relatively straightforward.

Anti-Symmetry: Consider relations r_a to be anti-symmetric, so that, $(e_1, r_a, e_2) \implies not\,(e_2, r_a, e_1)$ We can encode this in our likelihood term with orthagonal projectors for the head and tail, i.e., $\vec{\mathbf{p}}_1^r \perp \vec{\mathbf{p}}_2^r$ so that we take the orthagonal projections of the head and tail entity when the direction of the relation is reversed.

Inversion: Consider relations r_a, r_b to be inversions of each other, so that, $(e_1, r_a, e_2) \implies (e_2, r_b, e_1)$ We can encode this in our likelihood term by switching the head and tail projectors, i.e., $\vec{\mathbf{p}}_1^{r_a} = \vec{\mathbf{p}}_2^{r_b}$ and $\vec{\mathbf{p}}_2^{r_a} = \vec{\mathbf{p}}_1^{r_b}$. It is easy to verify that this would result in $\mathcal{L}(\vec{\mathbf{e}}_1, r_a, \vec{\mathbf{e}}_2) = \mathcal{L}(\vec{\mathbf{e}}_2, r_b, \vec{\mathbf{e}}_1)$ which results in the desired inversion.

Composition: Consider relations r_c to be composed of r_a and r_b, so that, (e_1, r_a, e_2) *and* $(e_2, r_b, e_3) \implies (e_1, r_c, e_3)$ We can encode this in our likelihood terms with the following simple switch, i.e., $\vec{\mathbf{p}}_1^{r_c} = \vec{\mathbf{p}}_1^{r_a}$ and $\vec{\mathbf{p}}_2^{r_c} = \vec{\mathbf{p}}_2^{r_a}$. This would transitively align the composed relation with the head and tail entities e_1 and e_3.

Finally, we also add a identity-matrix scaling factor to the relation projectors so that we can retain a percentage of the original node embedding dimensions in the projected versions as well:

$$\mathcal{L}(\vec{\mathbf{e}}_1, r, \vec{\mathbf{e}}_2) = sim((\vec{\mathbf{e}}_1^T \otimes (\vec{\mathbf{p}}_1^r + s\mathbb{I}))(\vec{\mathbf{e}}_2^T \otimes (\vec{\mathbf{p}}_2^r + s\mathbb{I})) \tag{3}$$

While the notion of head and tail projectors is also present in the TransD [12] model, our similarity function, which is just a dot product, enables block-sampling and optimization advantages. As a result, our model is scalable with the block optimizations proposed by Lerer et al. [18] and expressive. Our model's scalable and expressive nature forms an adaptive and scalable base embedding space to enable the multi-task and multi-modal transfer of knowledge from task-specific models.

4.2 Multi-Task Augementation

We briefly revisit our definition of task models and provide a simplified abstraction for the rest of the paper. Consider a consider discrete prediction models $\mathcal{M}^j$ to take an input entity $e_1^j \in \mathbf{E}_1^j$ from the knowledge graph, and produce a predicted output entity $e_2^j \in \mathbf{E}_2^j$, thus inducing a connection across the two entity sets $\mathbf{E}_1^j, \mathbf{E}_2^j \in \mathcal{E}$. Note that the specific prediction task j modeled by $\mathcal{M}^j$ may vary even between the same pair of entity sets $\mathbf{E}_1^j, \mathbf{E}_2^j \in \mathcal{E}$. For instance, a POI recommendation model could connect the user entities to the preferred venue nodes in our knowledge graph. Simultaneously, a location-based recommender could produce a different map between the same two entity sets, since it has a different objective function. Thus each task-model generates a different mapping between the two spaces, depending on its inductive bias and task objective. We will refer to these mappings as task-specific counterfactual links since they may not exist in the knowledge graph but are rather predicted by the task-model. For simplicity, we only consider task-models that establish connections between pairs of entity sets, although the proposed framework is general to multi-variate scenarios and regression with appropriate discretization.

4.2.1 Viewing Task-Models as Interventions. Our key insight is to consider each task-specific model as an intervention on a specific subset of nodes in the knowledge graph, analogous to a medical treatment applied to a patient. Note that the specific intervention depends on both the task (or objective) of the trained model and the model architecture, i.e., its inductive bias. Our key objective in the rest of this section is to develop a consistent pathway to enrich or complete the knowledge graph based on the intervention biased predictions. Clearly, these predictions should not be treated equivalently. Some task-models are likely to predict better the existence of certain types of links in the original knowledge graph. We refer to these intervention biased links as counterfactual links, in contrast to the factual links that exist in the knowledge graph. The counterfactual links may signal the existence of a factual link, depending on the applied intervention. We also note that all of the data used to train the task-models may already be present in the knowledge graph (e.g., a subset of the links in the graph), and does not necessarily involve any additional features or node attributes.

4.2.2 Intervention-Biased Counterfactual Links. Links in the base knowledge graph are the factual links across entities. At the same time, we refer to the links suggested by the task models as counterfactual links, from the intervention perspective. Specifically, task model $\mathcal{M}^j$ takes an input entity $e_1^j \in \mathbf{E}_1^j$ and predicts the output entity $e_2^j \in \mathbf{E}_2^j$ under task j. We refer to this as the primary counterfactual link, (e_1^j, e_2^j), since it is directly predicted by model $\mathcal{M}^j$. In contrast to the factual links across entities in the knowledge graph, the counterfactual link suggested by the model is intervention-biased. Thus, to employ the model predicted edges to enrich the knowledge graph, we need to develop a bias-elimination procedure that can extract the relevant information from each task-model and enrich the knowledge graph.

4.2.3 Cross-Modal Counterfactual Links. Let us revisit the task model, $\mathcal{M}^j$, which predicts intervention-biased counterfactual links of the form (e_1^j, e_2^j), where $e_1^j \in \mathbf{E}_1^j$ and $e_2^j \in \mathbf{E}_2^j$. We can evaluate additional counterfactual links by sampling nodes in the one-hop neighborhood of e_2^j and connecting them to e_1^j, and vice-versa. This enables us to infer connections across entities in two different correlated tasks and permit models to model transfer learning. Note that the counterfactual links may not have any associated relation, while the knowledge graph's factual links always do.

4.3 Creating Intervention-Biased Counterfactual Links

Each node in the knowledge graph is updated via the feedback from the external prediction models. The models supply intervention-biased counterfactual edges for each node, and these edges are appropriately leveraged to update the underlying node embeddings of the knowledge graph nodes.

We now present the knowledge transfer to node embeddings from the perspective of one specific focus node, $\mathbf{e}_i \in \mathbf{E}_i$. Consider task models $\mathcal{M}_j : \mathbf{E}_1^j \rightarrow \mathbf{E}_2^j$ following our notations from the previous subsection, i.e., different task-models with the input entity set $\mathbf{E}_i$ in which the focus node $\mathbf{e}_i$ lies. Each task model now produces an output prediction corresponding to the input $\mathbf{e}_i$. Consider the following set of predicted outputs from the task models:

- $e_i \rightarrow \mathcal{M}_{j1} \rightarrow e_{j1} \in \mathbf{E}_{j1}^2$
- $e_i \rightarrow \mathcal{M}_{j2} \rightarrow e_{j2} \in \mathbf{E}_{j2}^2$
- $e_i \rightarrow \mathcal{M}_{j3} \rightarrow e_{j3} \in \mathbf{E}_{j3}^2$

Each task may provide a distinct modality of edge connections, i.e., the sets $\mathbf{E}_{j1}^2$, $\mathbf{E}_{j2}^2$ and $\mathbf{E}_{j3}^2$ may be distinct from each other.

Let us now consider the set of intervention-biased link inferences we can make from these predictions. First, the primary links, namely (e_i, e_{j1}), (e_i, e_{j2}), and (e_i, e_{j3}). However, our inferences are not limited to these primary links alone. Let us now consider the set of 1-hop neighbors of each of the predicted primary connections, i.e., the 1-hop neighborhood $\mathcal{N}_{j1}$ for e_{j1}, $\mathcal{N}_{j2}$ for e_{j2} and $\mathcal{N}_{j3}$ for e_{j3}. Further, let us denote these one-hop neighbors as $n_{j1} \in \mathcal{N}_{j1}$, $n_{j2} \in \mathcal{N}_{j2}$ and $n_{j3} \in \mathcal{N}_{j3}$. These neighboring nodes can belong to different entity sets which may not be the same as $\mathbf{E}_{j1}^2$, $\mathbf{E}_{j2}^2$ or $\mathbf{E}_{j3}^2$. As a result, these transitive 1-hop connections result in several new cross-modal inferences in addition to the primary counterfactual links. These cross-modal links are also intervention-biased by association to the primary counterfactuals,(e_i, e_{j1}), (e_i, e_{j2}) and (e_i, e_{j3}), from which

they originate. We refer to this new transitive set of cross-modal links as the set of secondary counterfactual links.

Thus, for each predictive model, namely $\mathcal{M}_{j1}$, $\mathcal{M}_{j2}$, and $\mathcal{M}_{j3}$, we obtain sets of both primary and secondary intervention-biased counterfactual links for the focus entity e_i, which we can leverage to update the respective knowledge graph node embeddings. We will refer to the sets of counterfactual links for these three models, with the focus entity e_i as $C_{j1}(e_i)$, $C_{j2}(e_i)$, and $C_{j3}(e_i)$ respectively.

4.3.1 Modular Interface with the Task-Models. The overall modularity of our knowledge space representation approach derives from the formulation of the above edge likelihood functions, purely dependent on the set of entities, entity types, relation types of links, and the specific links in the knowledge graph, while being completely agnostic to how the links are added or suggested by the task models. The key intuition is to maintain an independent representation space, which is updated by external models by leveraging their counterfactual edge suggestions. We thus keep a consistent interface between the knowledge representation space and all external task models.

4.4 Individualized Treatment Effect for KG Nodes

We now briefly talk about our perspective to leverage the set of counterfactual links for node embedding updates. We closely follow the Rubin-Neyman causal model in our description [15, 29]. Let us consider the set of factual links of the focus entity $\mathbf{e}_i \in \mathcal{E}_i$, that are present in the knowledge graph, and are known with certainty.

Consider a specific factual link (e_i, r_F, e_j). Under our base embedding model, the likelihood of this factual link is given by:

$$\mathcal{L}(\vec{\mathbf{e}}_1, r, \vec{\mathbf{e}}_2) = sim((\vec{\mathbf{e}}_1^T \otimes (\vec{\mathbf{p}}_1^r + s\mathbb{I}))(\vec{\mathbf{e}}_2^T \otimes (\vec{\mathbf{p}}_2^r + s\mathbb{I})) \tag{4}$$

How do we then estimate the likelihood of a counterfactual link? A simple approach is to estimate the likelihood of any counterfactual link, (e_i, r_{CF}, e_j) suggested by a task model using the same base likelihood model:

$$\mathcal{L}(\vec{\mathbf{e}}_1, r_{CF}, \vec{\mathbf{e}}_2) = sim((\vec{\mathbf{e}}_1^T \otimes (\vec{\mathbf{p}}_1^{r_{CF}} + s\mathbb{I}))(\vec{\mathbf{e}}_2^T \otimes (\vec{\mathbf{p}}_2^{r_{CF}} + s\mathbb{I})) \tag{5}$$

The counterfactual relation-type r_{CF} (of the counterfactual link) *may or may not* be predicted by the external model. We provide three heuristics to address the case where r_{CF} is not known:

Relation-Agnostic (RA) Counterfactual Likelihood:

$$\mathcal{LRA}^{CF}(\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2) = ||\vec{\mathbf{e}}_1 \otimes \vec{\mathbf{e}}_2|| \tag{6}$$

where $||$ denotes a suitable norm function such as L2 distance or hinge loss.

The intuition of $\mathcal{LRA}^{CF}$ is to maximize the dimensions along which the two entity embeddings match. This strategy effectively increases the likelihood of any valid relation-type between the entity pair depending on the projection component, as long as the different relation-types are not anti-correlated.

Preferred-Relation (PR) Counterfactual Likelihood:

$$\mathcal{LPR}^{CF}(\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2) = \underset{r|\mathbf{R}_r:\mathbf{E}_1^r \rightarrow \mathbf{E}_2^r}{\text{argmax}} \sigma((\vec{\mathbf{e}}_1 \otimes \vec{\mathbf{p}}_1^r).(\vec{\mathbf{e}}_2 \otimes \vec{\mathbf{p}}_2^r)) \tag{7}$$

$\mathcal{LPR}^{CF}$ only considers the most-likely relation-type for any pair of entities in the likelihood estimation. This formulation is much more reliable than $\mathcal{LRA}^{CF}$ for pairs of entity types that have anti-correlated relations between them.

Relation-Sum (RS) Counterfactual Likelihood:

$$\mathcal{LRS}^{CF}(\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2) = \sum_{r|\mathbf{R}_r:\mathbf{E}_1^r \rightarrow \mathbf{E}_2^r} \sigma((\vec{\mathbf{e}}_1 \otimes \vec{\mathbf{p}}_1^r).(\vec{\mathbf{e}}_2 \otimes \vec{\mathbf{p}}_2^r)) \tag{8}$$

$\mathcal{LRS}^{CF}$ amortizes the gradients across all the relation-types between any pair of entity sets.

However, more fundamentally, all the above likelihoods assume that every suggested counterfactual link is well aligned to the factual set of links of each node. This may not hold if the set of task-models are trained with very distinct training objectives, or they learn fundamentally different views of the underlying node space owing to either their inductive biases or the training objective. In such a case, the counterfactual likelihood, $\mathcal{L}^{CF}(\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2)$ (where $\mathcal{L}$ is *LRS*, *LPR*, or *LRA*), must account for the biases introduced by the specific task model into the counterfactual links suggested by it. This equation would hold even if the counterfactual links have specific relation types r_{CF} associated with them. We propose to view this bias as a fundamental distributional shift on the latent node features (note that in the absence of node attributes, the latent node features only include the node embeddings learned by our base embedding model via Equation (3)). This view is grounded in the notion of individualized treatment effect [15], wherein we evaluate the effect of an intervention on a specific node. More concretely, let us consider the base embeddings of entity set $\mathbf{E}_i$ to be drawn from a factual distribution (that satisfies the factual likelihood $\mathcal{L}(\vec{\mathbf{e}}_1, r_F, \vec{\mathbf{e}}_2)$) in Equation (3) as follows:

$$\begin{aligned} \vec{\mathbf{e}}_1 &\sim P^F(\mathbf{E}_1^{r_F}) \\ \vec{\mathbf{e}}_2 &\sim P^F(\mathbf{E}_2^{r_F}) \end{aligned} \tag{9}$$

Conversely, the node embeddings that satisy the counterfactual links drawn from each task-model $\mathcal{M}^j$, as described in the previous subsection, induce a different distribution in the embedding space for the same nodes, $\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2$, depending on the objectives and inductive biases of model $\mathcal{M}^j$:

$$\begin{aligned} \vec{\mathbf{e}}_1 &\sim P_j^{CF}(\mathbf{E}_1^j) \\ \vec{\mathbf{e}}_2 &\sim P_j^{CF}(\mathbf{E}_2^j) \end{aligned} \tag{10}$$

The above mismatch results in a clearly quantizable distributional difference in the factual and counterfactual embedding distributions for each task model $\mathcal{M}^j$ for each entity set $\mathbf{E}$, which can be given by:

$$\Delta_j(P^F(\mathbf{E}), P_j^{CF}(\mathbf{E})) \tag{11}$$

We now show how to learn this distributional difference via counterfactual residuals, so that we can transfer knowledge between the node embeddings and the respective task models.

4.5 Eliminating Distributional Biases via Residual Learning

As described in the previous subsection, all versions of the counterfactual likelihood are prone to the distributional mismatch problem that we presented in the previous subsection. This implies that the updates that are obtained by optimizing any of the above three

objectives, namely $\mathcal{LRA}^{CF}$, $\mathcal{LPR}^{CF}$ and $\mathcal{LRS}^{CF}$ are likely to differ depending on the task model $\mathcal{M}^j$ that was used to create the counterfactual links. As a result, we must learn the correspoding distributional differences and account for them when we update the underlying knowledge graph embeddings.

This is a manifestation of a covariate shift, a special case of domain adaptation [6] and can be addressed via residual learning. Specifically, we can zoom into the respective counterfactual link likelihoods:

$$\mathcal{LPR}^{CF}_{(e_1,e_2)} = \underset{r|\mathbf{R}_r:\mathbf{E}_1\rightarrow\mathbf{E}_2}{\text{argmax}} \ \sigma((\vec{\mathbf{e}}_1 \otimes \vec{\mathbf{p}}^r_1).(\vec{\mathbf{e}}_2 \otimes \vec{\mathbf{p}}^r_2)) \tag{12}$$

We must view the above equation as transforming a domain-adapted or intervention-adapted variation of the node embedding instead of the direct node embedding. Under this consideration, the intervention adapted embeddings can be given as follows:

$$\begin{aligned} \vec{\mathbf{e}}^j_1 &= \vec{\mathbf{e}}_1 + \delta^j_{\mathbf{E}_1}(\vec{\mathbf{e}}_1) \\ \vec{\mathbf{e}}^j_2 &= \vec{\mathbf{e}}_2 + \delta^j_{\mathbf{E}_2}(\vec{\mathbf{e}}_2) \end{aligned} \tag{13}$$

Here, we learn the residual functions $\delta^j_{\mathbf{E}_1}$, $\delta^j_{\mathbf{E}_2}$ to optimize for the components of the underlying embeddings of both the entity sets $\mathbf{E}^j_1, \mathbf{E}^j_2$ that are best captured by the counterfactual links from model $\mathcal{M}^j$, and maximize separation for the embedding dimensions that are inversely impacted by the model intervention. Note that the residual functions $\delta^j_{\mathbf{E}}$ are specific to both the model $\mathcal{M}^j$ as well as the entity set $\mathbf{E}$, since each entity set is impacted differently by the model intervention.

The LPR likelihood term, $\mathcal{LPR}^{CF}_{(\vec{\mathbf{e}}_1,\vec{\mathbf{e}}_2)}$ from Equation (7) can then be viewed as optimizing the bias-adapted version of the focus node embeddings as well as its connections. In other words, we are now optimizing $\mathcal{LPR}^{CF}_{(\vec{\mathbf{e}}^j_1,\vec{\mathbf{e}}^j_2)}$ where $\vec{\mathbf{e}}^j_1 = \vec{\mathbf{e}}_1 + \delta^j_{\mathbf{E}_1}(\vec{\mathbf{e}}_1)$ is backpropagated instead of $\vec{\mathbf{e}}_1$, and likewise for $\vec{\mathbf{e}}_2$.

Thus, to learn the above residual function $\delta^j_{\mathbf{E}}$, we need to learn the embedding dimensions that must be bias-corrected to account for the intervention-bias of model $\mathcal{M}^j$. We use a single learnable neural network layer for each residual function $\delta^j_{\mathbf{E}}$. For our experiments, we used a linear form with *tanh* activation.

$$\delta^j_{\mathbf{E}_1}(\vec{\mathbf{e}}_1) = tanh(\mathbf{W}^j_{\mathbf{E}_1}\vec{\mathbf{e}}_1 + \mathbf{b}^j_{E_1}) \tag{14}$$

The motivation for the above from of δ is grounded in the notion of decreasing returns - We hypothesize the model interventions to produce a scaled effect on the distributional characteristics of each node, which can be captured by the matrix $\mathbf{W}^j$, while the tanh function changes the shape of the corresponding dimension-discount, i.e., the scaling factor for each dimension is a tanh shaped learnable receptive curve.

We describe our training objectives to learn all the parameters in the next section.

4.6 Discrepancy Distance Regularization

We do not make any assumptions about the underlying feature distributions of nodes in the knowledge graph. Given the set of node attributes, or in the case of purely structural embeddings like ours, we aim to learn the set of features that vary across the factual and counterfactual distributions with our task-specific and entity-type specific residual functions, $\delta^j_{\mathbf{E}}$, in order to obtain the counterfactual views of each node embedding:

$$\vec{\mathbf{e}}^j_1 = \vec{\mathbf{e}}_1 + \delta^j_{\mathbf{E}_1}(\vec{\mathbf{e}}_1) \tag{15}$$

In other words, residuals that reduce the discrepancy between the feature distributions (i.e., the node embeddings) across the factual and counterfactual domains should not bias to "unreliable" features of the node embeddings. For example, if a recommendation model heavily favors high degree item nodes, the resulting residual function $\delta^j_{\mathbf{E}_1}(\vec{\mathbf{e}}_1)$ where $\mathbf{E}_1$ is the item set, may attempt to completely eliminate this feature difference. As a result, the underlying item node embeddings $\vec{\mathbf{e}}_1$ may not learn any informative covariates from the counterfactual links, if all the difference is explained by the residuals alone.

To prevent this degenerate solution, where the node embedding does not receive any informative task-model updates, we leverage the notion of discrepancy distance. This distance measure serves as a strong regularizer to prevent overfitting the learned residual to any specific node feature, much akin to the discrepancy distance minimizers proposed in [15, 22]. As a result, components of the counterfactual distribution can partially flow to the base node embedding $\vec{\mathbf{e}}_1$.

Past work has shown how to derive generalization bounds with a well-defined empirical discrepancy distance for a limited hypothesis class of models [22]. In our work, however, we do not restrict the set of task-models to any specific hypothesis class, but instead propose a non-linear L1 norm on the residual value across all task-models, with a task and entity type specific weight factor $\mathbf{W}^j_{\mathbf{E}}$ as follows:

$$disc(\delta^j_{\mathbf{E}_1}(\mathbf{e}_1)) = ||\sigma(\mathbf{W}^j_{\mathbf{E}_1} \times \delta^j_{\mathbf{E}_1}(\mathbf{e}_1))|| \tag{16}$$

We now describe how the above components fit into an overall training strategy for graph-to-model and model-to-graph co-training, and also model-to-model (via graph) cross-training.

5 TRAINING METHOD

In this section, we describe the overall training objective and the procedures for simultaneous graph embedding updates and model training and the transfer of knowledge from one model to a different model by updating the respective node embeddings.

5.1 Learning the Bias-Elimination Residuals

As described previously, the objective of the bias-elimination residual is to learn the distributional differences in the node embeddings across the original embedding space and the transformed embedding space of each focus entity $e_1 \in \mathbf{E}^j_1$, which has a primary counterfactual link to $e_2 \in \mathbf{E}^j_2$, the prediction output of model $\mathcal{M}^j$, and secondary counterfactual links to the neighbors of e_2, namely $\mathcal{N}_{e_2}$, as described in Section 4.3.

For a given randomly sampled subset of focus entites, $\mathbf{S}_1 \subseteq \mathbf{E}^j_1$, we can generate all the possible primary and secondary counterfactual links for each focus entity $e_1 \in \mathbf{S}_1$ using the model $\mathcal{M}^j$ as described in Section 4.3. Let us denote this set of links as $(e_1, e_{CF}) \in \mathbf{CF}^j(\mathbf{S}_1)$. Similarly, we also collect the set of factual links associated with the same subset of focus entities $\mathbf{S}_1$, which

we denote as $(e_1, r, e_F) \in \mathbf{F}(\mathbf{S}_1)$. While the factual links are specific to the focus entity set $\mathbf{E}_1^j$, the counterfactual links are specific to the task-model subscript j and the focus entity set $\mathbf{E}_1^j$.

In order to learn the distributional difference residual function δ_j across the counterfactual and factual link sets, we optimize the following two stochastic objective functions alternatingly (the stochasticity emerges from the random selection of the focus entity subset, $\mathbf{S}_1 \subseteq \mathbf{E}_1^j$):

$$\mathcal{L}_F = \sum_{(e_1,r,e_F)\in \mathbf{F}(\mathbf{S}_1)} sim((\vec{\mathbf{e}}_1^T \otimes (\vec{\mathbf{p}}_1^r + s\mathbb{I}))(\vec{\mathbf{e}}_F^T \otimes (\vec{\mathbf{p}}_2^r + s\mathbb{I})) \tag{17}$$

$$\mathcal{L}_{CF}^j = \sum_{(e_1,e_{CF})\in \mathbf{CF}^j(\mathbf{S}_1)} \mathcal{LPR}^{CF}(\vec{\mathbf{e}}_1 + \delta^j_{\mathbf{E}_1}(\vec{\mathbf{e}}_1), \vec{\mathbf{e}}_{CF} + \delta^j_{\mathbf{E}_{CF}}(\vec{\mathbf{e}}_{CF})) + d1 \times disc(\delta^j_{\mathbf{E}_1}(\vec{\mathbf{e}}_1))) + d2 \times disc(\delta^j_{\mathbf{E}_1}(\vec{\mathbf{e}}_i)) \tag{18}$$

Note that $\mathbf{E}_{CF}$ can be any entity set, and is not limited to just $\mathbf{E}_2^j$, since we also used the one-hop neighbors of e_2 in the counterfactual set. While Equation (18), uses the Preferred-Relation (PR) Counterfactual Likelihood from Equation (7), we could also choose to apply the Relation-Agnostic (RA) version (Equation (6)) or the Relation-Sum (RS) version (Equation (8)).

Optimizing Equation (17) and Equation (18) alternatingly results in stochastic updates to the node embeddings $\vec{\mathbf{e}}_i$ of entities in the focus set $\mathbf{S}_{\mathbf{E}_i}$, the residual functions δ_j, and the discrepancy distance measures, scaled by head and tail parameters $d1, d2$.

5.2 Graph and Model Co-Training

The previous technique works for one direction of knowledge transfer, namely model-to-graph. It is possible to train node embeddings and task-models bi-directionally if task-model is a white-box model with a continuous differentiable objective function.

Note that each residual function is applied additively to the node embeddings with the $\mathcal{LPR}^{CF}$ likelihood when we wish to update the node embeddings and residual functions, as described in Equation (18). However, in Equation (18), the model is held fixed, i.e., the backpropagation updates are only carried out to the node embeddings. The direction of information flow is from the task-model to the node embeddings.

Conversely, if we wish to update the task model $\mathcal{M}^j$, we need the gradients to flow from the node embeddings to the model rather than the reverse. To achieve this effect, we can apply the same residual transformations to the node embeddings of factual links that are present in the graph (instead of the counterfactual links), and add them as a soft-alignment criterion to the model optimization objective. Towards this goal, we again create a set of factual links of focus entities $\mathbf{S}_1 \subseteq \mathbf{E}_1^j$, namely $\mathbf{F}(\mathbf{S}_1)$ as described in Section 5.1. We can then apply the following soft alignments ($\mathcal{SA}^j(e_1, e_F)$) for the entity pairs in each link $\sum_{(e_1,r,e_F)\in \mathbf{F}(\mathbf{S}_1)}$:

$$\mathcal{SA}^j(e_1, e_F) = \mathcal{L}(\vec{\mathbf{e}}_1 + \delta^j_{\mathbf{E}_1}(\vec{\mathbf{e}}_1), \vec{\mathbf{e}}_{CF} + \delta^j_{\mathbf{E}_{CF}}(\vec{\mathbf{e}}_{CF})) \tag{19}$$

Where $\mathcal{L}$ denotes the same basic factual likelihood equation described in Equation (3).

Let the white-box model $\mathcal{M}^j$ have a differentiable objective function O^j, as a function of it's predicted outputs, i.e., the primary counterfactual links. We can now add the following term to it:

$$\tilde{O}^j = O^j + \lambda^j \sum_{(e_1,r,e_F)\in \mathbf{F}(\mathbf{S}_1)} (\mathcal{SA}^j(e_1, e_F) - \mathcal{M}^j(e_1, e_F)) \tag{20}$$

Here, we are overloading the $\mathcal{M}^j(e_1, e_F)$ term to indicate how the model measures the proximities of its input and output entities. Note that the second term here makes the model align its own proximities to what is suggested using the residual functions and node emebddings. The parameter λ^j determines the strength of the regularization applied to the model.

We do not need to have any residual discrepancies here since we are not updating the residual parameters, but rather only the model parameters of task-model $\mathcal{M}^j$. We also permit an overall hybrid update model where we simultaneously change the model parameters as well as the node and residual functions by iteratively and alternatingly optimizing all three objectives at the same time, namely: Equation (17), Equation (18), and Equation (20).

5.3 Model to Model Cross-Training

Let us consider the following directionality of model-to-model cross-training: say $\mathcal{M}^{j_1} \rightarrow \mathcal{M}^{j_2}$. For cross-training two models we need the condition $\{E_1^{j_1}, E_2^{j_1}\} \cap \{E_1^{j_2}, E_2^{j_2}\} \neq \Phi$ to hold. Note that this condition simply states that at least one of the entity sets whose node emebddings are updated by the counterfactual likelihoods in Equation (18) must be present across both the models.

We explain the model to model cross-training scenario with a sample scenario where $E_2^{j_1} = E_1^{j_2}$.

- Learn the first cut set of node embeddings by leveraging the factual links in the graph for all the entity sets, and optimizing for the likelihood given by Equation (3).
- Select the first model $\mathcal{M}^1$ in the above sequence and learn the counterfactual residuals by alternating optimization with the likelihoods in Equation (17) and Equation (18).
- In each prediction with a focus entity, i.e., $e_1 \in \mathbf{E}_1^{j_1}$, connect the predicted output entity of $\mathcal{M}^{j_1}$, say $e_2 \in \mathbf{E}_2^{j_2}$, to all the one-hop neighbors of e_1, and use it to update the node embeddings while holding the residual functions learned above as constant.
- Finally, with the node embeddings updated by $\mathcal{M}^1$, perform the graph-to-model updates as described in Section 5.2 in order to improve the performance of model $\mathcal{M}^2$.

We observe that our overall framework is not theoretically exchangable since the order in which the models are cross-trained with the knowledge graph also influences the final results. This is a fundamental limitation of the causal interpretation of node embeddings and counterfactual updates since we need to assign a directionality of cause to effect. This limitation also applies to the order in which models are co-trained and updated with the knowledge graph in our framework.

6 EXPERIMENTAL RESULTS

In this section, we present our experimental analyses on diverse multi-domain datasets and validate our framework.

First, we show that counterfactual enrichment with effective task-models can significantly improve the quality of node embeddings for modalities with sparse connections by evaluating the

updated embeddings on the held-out link completion task. Next, we show that co-training a context-aware neural recommendation model with the knowledge graph leads to simultaneous embedding updates and better model performance for nodes with lower degrees, although we notice a small degradation in the performance for high-degree nodes. Additionally, we exhibit that we can significantly improve the above context-aware neural recommendation model by leveraging a distributed word embedding model using the illustrated cross-training method. Finally, we do a scalability analysis against publicly available baseline implementations, and conclude with limitations and discussion.

6.1 Data Description, Setup

Google Local Reviews Dataset [9, 28]: Users rate businesses on a 0-5 scale with temporal, spatial, and textual context available for each review. We filter this dataset with a criteria of atleast 10 users per business and 5 businesses per user recursively, and eliminate all reviews with less than 3-star rating. The resulting dataset has 38,614 users and 26,922 businesses, and the following contextual node types:

- Review Words
- Business Name Words
- Categories of the Business
- Price
- Location nodes - states, cities
- Temporal - time (binned inti 6-hour chunks), month, day

We create our knowledge graph by connecting all users to the businesses they rated, the name and review words of the businesses to each business, the review words, categories of visits and business names to the users who rated them, the priceiness, locations and times to businesses and users. On each of these links, we associated a 1-4 level depending on the strength of the associations (measured statistically on a per-user and per-business basis). This constitutes our relation types.

Entity Type	Count
Users	38,614
Businesses	26,922
Business Name Words	2,000
Review Words	5,000
Business Categories	650
Priciness	4
Time	23
Location	312
Total Links	7,325,614

Table 1: Google Local KG Statistics

Yelp Challenge Dataset: Users rate businesses on a 0-5 scale with temporal, spatial, and textual context available for each review. We filter this dataset with a criteria of at least 30 users per business and 10 businesses per user recursively, and eliminate all reviews with less than 3-star rating. The resulting dataset has 25,3695 users and 69,738 businesses, and we obtain the following contextual nodes:

- Users
- Restaurants
- Review Words
- Business Attributes
- Location nodes - states, cities, lat-lon (binned using a KD-tree)
- Temporal - time (binned by 6-hour chunks), month, day

We create our knowledge graph by connecting all users to the restaurants they rated, the review words and attributes of the restaurants to each restaurant, the location nodes, the associated time nodes, and likewise for the users as well. On each of these links, we associated a 1-4 level depending on the strength of the associations (measured statistically on a per-user and per-business basis). This constitutes our relation types.

Entity Type	Count
Users	20,750
Restaurants	75,871
Review Words	2,000
Business Attributes	200
Time	23
Location	1,062
Total Links	10,102,877

Table 2: Yelp KG Statistics

Baselines: We used the following knowledge graph embedding baselines as a representative set to evaluate the edge completion task: TransE [2], DistMult [36], ComplEx [32], Rotate [30]. We used the OpenKE implementations[3] in Tensorflow/PyTorch with default parameter settings, wherever applicable.

6.2 Task-Models

For both datasets, we used a pair of task models that both have the same input entity-set (users), and different output entity sets (business category and businesses respectively).

We train the distributional word2vec word-embedding model [23] on the set of review text words, business names and all the business attributes text over all reviews in the dataset. We use the basic version (non-transfer) of the context-aware recommender proposed in Krishnan et al. [16] with the non-textual categorical links of the users and businesses (as above) forming the context of each review. To predict business category/attribute words for each user, we take an average of their review word set embeddings, and map the average to the closest business category words as learned by the model. Note that to train the word2vec model, we use the review text as context for the business attribute text.

Parameters: In both the above datasets, for the context-aware recommendation model [16], we use the author recommended parameters with 200-dimensional embeddings, while we use the gensim[4] implementation of word2vec with a maximum 10-length window. The additional parameters of our model, such as the discrepancy scaling in Equation (18) were tuned with an exponential grid-search approach (e^{-5} to e^{0}). The knowledge graph and counterfactual residuals were also trained with 200-dimensional embeddings, and implemented in Tensorflow, and run on a Tesla K80 GPU.

[3] http://139.129.163.161//
[4] https://pypi.org/project/gensim/

Link Type	User to Business		User to Category	
Metric	**R @ 5**	**R @ 10**	**R @ 5**	**R @ 10**
TransE	0.43	0.60	0.52	0.68
RotatE	0.59	0.72	0.65	0.80
DistMult	0.56	0.70	0.63	0.77
CompleX	0.57	0.70	0.61	0.76
MutatE-F	0.58	0.73	0.64	0.79
MutatE-CF	**0.62**	**0.80**	**0.68**	**0.84**

Table 3: Overall Link Prediction Results

Metrics for Link Prediction: In both the datasets, we attempt to predict held-out links using the embeddings learned by our models, as well as the embedding baselines. For each held-out link of the form (e_1, r, e_2), we create several negative samples of the form $(e_1, r, \tilde{e}_2)$ and $(\tilde{e}_1, r, e_2)$, i.e., with the same relation type and head and tail entity types, however a randomly sampled entity for either the head or tail. We then rank the entire list of negative samples against the true link (e_1, r, e_2) under each embedding model, and measure the **Recall@K**, **NDCG@K** values of the respective ranked lists. Specifically, we measure the **Recall@5**, **Recall@10** for two types of heldout links - User → Business and User → Category word (Attribute in case of yelp), for a 100-length ranked list.

6.3 Primary Results - Link Prediction

The above two knowledge graphs are evaluated on our link completion task. We randomly tag 20% of the user nodes as held-out nodes. We then held out two types of links for these users - we held out half of their user-business links, and half of their user-business attribute/category word links. Note that these two link types directly correspond to the two task models we used: The word2vec model predicts user-business category word links and the context-aware recommender predicts the user-business links.

For our model, we present two variants - MUTATE-F, which only uses the factual nodes, and MUTATE-CF, which uses counterfactual enrichment for the held-out user set. Specifically, the top-5 words predicted by the word2vec model, and the top-5 businesses predicted by the recommender are used to form counterfactual user-business and user-word links. We also trained all the baseline embedding models on the same knowledge graphs, and attempted to predict the same set of heldout links using their trained embeddings.

Key Observations from Table 3: The relative order of performance of the baselines is roughly as expected, DistMult [36] performs somewhat weakly owing to the inverse nature of some relation-types in our graphs across user-context-business paths. In contrast, our base model is able to overcome this challenge, and performs comparably to the other baselines.

We also observe that our MUTATE-CF model strongly outperforms all the competing models on the User-Word link prediction and User-Business link prediction tasks. This is unsurprising, since the two external task models, namely word2vec and the context-aware recommender, are able to much better predict the missing links and enrich the graph, as compared to the heuristic or path-based link completion approach in the other baselines. It is easy to see how we are able to leverage the inductive biases of the specific models - While the word2vec model is able to interpret the distributional properties of the review text, the context-aware recommender leverages the multiplicative predictors from the context features. Also, note that these two models only use data that is already used to construct the Knowledge Graphs, and do not depend on any external sources of data.

λ^j	e^{-5}	e^{-4}	e^{-3}	e^{-2}	e^{-1}
Word2Vec	-5.6%	-1.3%	**+8.1%**	-4.9%	-18.6%
Context Recommender	+2.8%	-1.03%	**+5.4%**	-8.6%	-28.9%

Table 4: Co-Training Performance Gains against the Information-flow Parameter λ^j from Equation (18)

6.4 Co-Training Model with Graph

In this section, we describe our co-training approach for the recommender model with the knowledge graph. Specifically, we make predictions from these models for users, and use these counterfactual links to update the knowledge graph embeddings, as described in Equation (17), and simultaneously, we make predictions from the updated embeddings for the users, and use these to augment the loss function of the recommender model, as described in Equation (19). In this manner, we attempt to improve the model performance over just training the model in isolation.

Although we did not achieve a dramatic performance difference, we clearly observe that overregularizing the model, as well as underregularizing the model are suboptimal. In other words the co-training proceeds best, when the regularizer λ^j is set to an optimal balance.

The numbers in Table 4 indicate the best performance improvements we were able to achieve for the recommender model under different settings of λ^j. A higher value of λ^j meant that the recommender was more constrained by the knowledge graph, while a lower value meant that more information flows from the model to the graph. Thus, we need an ideal tradeoff between the forward and reverse information flow.

6.5 Cross-Training across Tasks

In this section, we describe our cross-training approach for the recommender model by leveraging the word2vec model. Specifically, we first train the word2vec model on the base data, then use it to update the knowledge graph embeddings using the model to graph knowledge transfer method described in Section 5.3, and finally use the reverse direction to generate additional regularization for the recommender model, i.e., knowledge now flows from the update graph to the recommender model. Thus, the overall direction of knowledge flow is as follows:

$$\mathcal{M}^{word2vec} \rightarrow \textit{Knowledge Graph} \rightarrow \mathcal{M}^{context\text{-}aware\text{-}recommender}$$

Since the review text is strongly informative of the user embeddings and is also connected to the business embeddings owing to their shared link structure, we were able to achieve noticeable

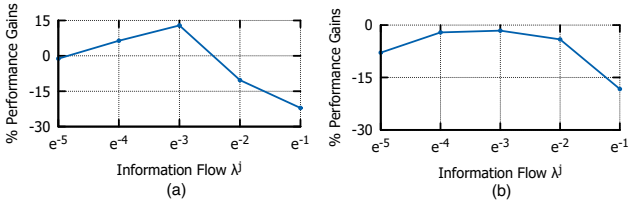

Figure 4: Cross-Training performance gains for the context-recommender with word2vec with respect to the parameter λ^j set to varying values as in Equation (18). Information flow directions are:
(a) $\mathcal{M}^{word2vec} \rightarrow KG \rightarrow \mathcal{M}^{context\text{-}aware\text{-}recommender}$ and
(b) $\mathcal{M}^{context\text{-}aware\text{-}recommender} \rightarrow KG \rightarrow \mathcal{M}^{word2vec}$.

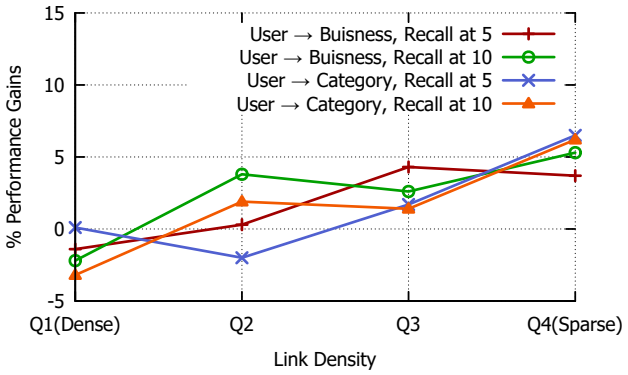

Figure 5: The gains of MUTATE-CF relative to MUTATE-F on the two types of link prediction. In each case, we measure the performance gains across 4 quartiles of users, arranged by the density of that specific type of link for the user.

performance gains for the recommender model (see Table 4) after we leveraged the sequence of update-steps described in Section 5.3.

6.6 Sparsity Analysis

In this subsection, we attempt to study the impact of counterfactual updates on sparse and non-sparse nodes. Specifically, for both the tasks, user-word link prediction and user-business link prediction, we study the relative gains obtained by counterfactual updates, i.e., the difference in performance of MUTATE and MUTATE-F for the different sparsity sets. $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$ and $\mathbf{Q}_4$ denote the four sparsity quartiles for each respective user node, and we then measure the average performance difference between MUTATE and MUTATE-F for each quartile in Figure 5.

As expected, the strongest gains are obtained for sparse users, i.e, users in quartiles Q3/Q4, since they lack the word-associations to help us learn better node embeddings. Thus, the distributional knowledge encoded in the word2vec model can significantly bridge this gap in the knowledge graph and enrich the corresponding node embeddings.

6.7 Limitations and Discussion

The two primary weaknesses of our work are non-exchangability of the order in cross-training, and the assumption of homoschedastic embeddings within each entity set. In other words, we assume that a single residual function, conditioned on the node embeddings of each node can fully account for the distributional differences introduced by the task-models.

A few alternatives exist to capture heteroschedastic node embeddings, such as gaussian mixture embedding spaces [4], but they are quite hard to implement efficiently within a knowledge graph neural network optimization framework. Further, since we do not bound the nature of the task-models, we do not have a tight bound to describe the discrepancy distance function in Equation (18). We leave these challenges as future work to study the trade-offs between generalizability and theoretical guarantees on the residual functions or overall exchangeability.

7 CONCLUSION

We propose a holistic view of knowledge graph and multi-task learning that permits the multi-directional transfer of knowledge between domain-specific knowledge graphs and task-models. The proposed framework is highly generalizable and can integrate diverse tasks and model architectures through a common set of underlying knowledge embeddings. Our framework can also model different task-specific distributions with the same underlying knowledge graph via counterfactual residual learning. We demonstrate the superiority of our solution over previous efforts. In the future, we intend to study the trade-offs between generalizability and theoretical guarantees on the residual functions and overall exchangeability.

REFERENCES

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* 11, 9 (2018).

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*.

[3] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The World Wide Web Conference (WWW)*.

[4] Francesco Paolo Casale, Adrian Dalca, Luca Saglietti, Jennifer Listgarten, and Nicolo Fusi. 2018. Gaussian process prior variational autoencoders. In *Advances in Neural Information Processing Systems*. 10369–10380.

[5] Dawei Cheng, Fangzhou Yang, Xiaoyang Wang, Ying Zhang, and Liqing Zhang. 2020. Knowledge Graph-based Event Embedding Framework for Financial Quantitative Investments. In *International Conference on Research and Development in Information Retrieval (SIGIR)*.

[6] Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research* 26 (2006).

[7] Patrick Ernst, Amy Siu, and Gerhard Weikum. 2015. Knowlife: a versatile approach for constructing a large knowledge graph for biomedical sciences. *BMC Bioinformatics* 16, 1 (2015).

[8] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2017. Knowledge graph embedding with iterative guidance from soft rules. *arXiv preprint arXiv:1711.11231* (2017).

[9] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 161–169.

[10] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *International Conference on Web Search and Data Mining (WSDM)*.

[11] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. 2003. Complete mining of frequent patterns from graphs: Mining graph data. *Journal of Machine Learning* 50, 3 (2003).

[12] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

[13] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge Graph Completion with Adaptive Sparse Transfer Matrix. In *International Conference on Artificial Intelligence (AAAI)*, Dale Schuurmans and Michael P. Wellman (Eds.).

[14] Yantao Jia, Yuanzhuo Wang, Hailun Lin, Xiaolong Jin, and Xueqi Cheng. 2016. Locally Adaptive Translation for Knowledge Graph Embedding. In *International Conference on Artificial Intelligence (AAAI)*.

[15] Fredrik Johansson, Uri Shalit, and David Sontag. 2016. Learning representations for counterfactual inference. In *International Conference on Machine Learning (ICML)*.

[16] Adit Krishnan, Mahashweta Das, Mangesh Bendre, Hao Yang, and Hari Sundaram. 2020. Transfer Learning via Contextual Invariants for One-to-Many Cross-Domain Recommendation. *arXiv preprint arXiv:2005.10473* (2020).

[17] Adit Krishnan, Ashish Sharma, and Hari Sundaram. 2018. Insights from the long-tail: Learning latent representations of online user behavior in the presence of skew and sparsity. In *International Conference on Information and Knowledge Management (CIKM)*.

[18] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. Pytorch-biggraph: A large-scale graph embedding system. *arXiv preprint arXiv:1903.12287* (2019).

[19] Ang Li and Judea Pearl. 2019. Unit Selection Based on Counterfactual Logic.. In *International Joint Conferences on Artificial Intelligence (IJCAI)*. 1793–1799.

[20] Wentian Li. 2002. Zipf's Law everywhere. *Glottometrics* 5 (2002), 14–21.

[21] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *International Conference on Artificial Intelligence (AAAI)*.

[22] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430* (2009).

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. [n.d.]. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*.

[24] Stephen L Morgan and Christopher Winship. 2015. *Counterfactuals and causal inference*. Cambridge University Press.

[25] Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Q. Phung. 2019. A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Association for Computational Linguistics, 2180–2189.

[26] Maximilian Nickel and Volker Tresp. 2013. An Analysis of Tensor Models for Learning on Structured Data. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) (Lecture Notes in Computer Science)*, Vol. 8189. Springer.

[27] Maximilian Nickel and Volker Tresp. 2013. Tensor Factorization for Multi-relational Learning. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) (Lecture Notes in Computer Science)*, Vol. 8190. Springer.

[28] Rajiv Pasricha and Julian McAuley. 2018. Translation-based factorization machines for sequential recommendation. In *International Conference on Recommender Systems (RecSys)*.

[29] Donald B Rubin. 1974. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology* 66, 5 (1974), 688.

[30] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* (2019).

[31] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *International Conference on Recommender Systems (RecSys)*.

[32] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML).

[33] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *International Conference on Knowledge Discovery & Data Mining (SIGKDD)*.

[34] Zhigang Wang, Juanzi Li, Zhichun Wang, Shuangjie Li, Mingyang Li, Dongsheng Zhang, Yao Shi, Yongbin Liu, Peng Zhang, and Jie Tang. 2013. XLore: A Large-scale English-Chinese Bilingual Knowledge Graph. In *International Semantic Web Conference (ISWC)*.

[35] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes.. In *International Conference on Artificial Intelligence (AAAI)*.

[36] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).