Multi-task Knowledge Graph Representations via Residual Functions

Adit Krishnan[1(✉)], Mahashweta Das[2], Mangesh Bendre[2], Fei Wang[2], Hao Yang[2], and Hari Sundaram[1]

[1] University of Illinois at Urbana-Champaign, Champaign, USA
{aditk2,hs1}@illinois.edu
[2] Visa Research, Palo Alto, CA, USA
{mahdas,mbendre,feiwang,haoyang}@visa.com

Abstract. In this paper, we propose MuTATE, a Multi-Task Augmented approach to learn Transferable Embeddings of knowledge graphs. Previous knowledge graph representation techniques either employ task-agnostic geometric hypotheses to learn informative node embeddings or integrate task-specific learning objectives like attribute prediction. In contrast, our framework unifies multiple co-dependent learning objectives with knowledge graph enrichment. We define co-dependence as multiple tasks that extract covariant distributions of entities and their relationships for prediction or regression objectives. We facilitate knowledge transfer in this setting: tasks→graph, graph→tasks, and task-1→task-2 via task-specific residual functions to specialize the node embeddings for each task, motivated by domain-shift theory. We show 5% relative gains over *state-of-the-art* knowledge graph embedding baselines on two public multi-task datasets and show significant potential for cross-task learning.

Keywords: Knowledge graphs · Knowledge graph embedding · Graph neural networks · Multi-task learning · Residual learning

1 Introduction

Knowledge graphs enable versatile storage, visualization, interpretation, and manipulation of large volumes of contextual information across interacting entities (nodes) via relations (links) in diverse domains such as linguistics (Wang et al. (2013)), biomedicine (Ernst et al. (2015)) and finance (Cheng et al. (2020)). The transitive entity association structure enhances inferencing applications involving entity attribute prediction and *entity-to-entity* relation prediction. However, the persistent challenges with knowledge graphs are two-fold, *link sparsity and its* task-unaware inflexible structure (Huang et al. (2019); Wang et al. (2014)). To overcome these challenges, a popular direction is to embed knowledge graphs in dense vector spaces (Bordes et al. (2013); Wang et al. (2014); Sun et al. (2019)) via path-based patterns such as *symmetry*, *anti-symmetry*, *composition* and *analogy* (Sect. 3.1). However, these learned patterns are static

J. Gama et al. (Eds.): PAKDD 2022, LNAI 13280, pp. 262–275, 2022.
https://doi.org/10.1007/978-3-031-05933-9_21

and not task-specific. To address this, the second direction integrates knowledge graph embeddings with specific learning tasks (Huang et al. (2019); Wang et al. (2019a)). In this case, the node/link embeddings are optimized for a single-task, but cannot combine or benefit multiple tasks.

Unlike these two directions, our approach unifies multi-task learning, graph enrichment, and embedding learning. We specifically focus on co-dependent tasks, i.e., tasks depending on shared aspects of the graph structure. As an example, we consider two well-defined prediction objectives in Fig. 1, book recommendation and book genre prediction. We consider a collaborative recommender model on the user-book links and a prediction model on the book-genre links.

These two task-models extract task-biased views of the knowledge graph depending on their *inductive biases*. However, both tasks (recommendation, genre prediction) require accurate book embeddings, i.e., shared subspace of the joint (*User*, *Item*, *Genre*) latent distribution. Further, each model can address link sparsity in the graph by predicting new links of the same type, thus transferring the extracted knowledge back to the graph. These newly predicted links represent the task-biased distribution learned by each model. Combining multiple tasks in this manner jointly enriches the graph as well as the other tasks through their shared subspaces. In summary, our contributions are as follows:

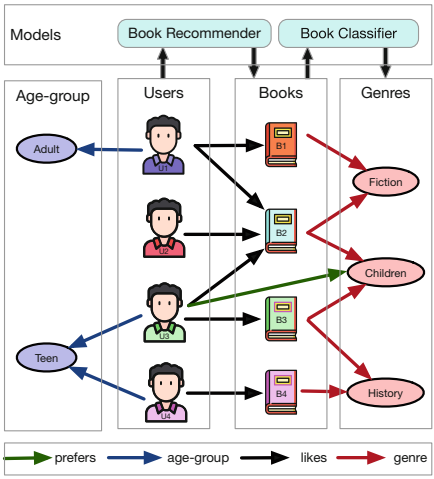

Fig. 1. Toy knowledge graph with four entity types: *users*, *books*, *age-groups*, *genres*. Entities are linked via user *prefers* genre, user *in* age-group, user *likes* book, book *is genre relations*. *Sample task-models include recommendation and book genre prediction.*

Merging Multi-task Learning and Knowledge Graph Embedding/Knowledge Graph Enrichment: We propose a holistic view of knowledge graphs and multi-task learning to enable bidirectional knowledge transfer between the graph and multiple co-dependent learning objectives.

Generalizability: The proposed framework makes no assumptions about the data-domain or learning tasks. We validate this empirically.

Modeling Multi-task Embedding Updates via Residuals: We identify the connection between multi-task knowledge graph updates and covariate shift (Johansson et al. (2016)) to unify multiple task distributions over shared node embeddings via task-specific residual functions.

Strong Experimental Results: We demonstrate strong experimental results on knowledge graphs constructed from two large public datasets, *Google Local*

Reviews[1] (He et al. (2017); Pasricha and McAuley (2018)) and *Yelp Challenge*[2] and using two co-dependent task-models, word2vec (Mikolov et al. (2013)) and a context-aware recommender (Krishnan et al. (2020)).

2 Problem Definition

Knowledge Graph Notations: We consider a heterogeneous directed knowledge graph with multiple entity (node) types, $\mathcal{E} = \{\mathbf{E}_1, \mathbf{E}_2 \cdots \mathbf{E}_{|\mathcal{E}|}\}$.

Factual Links: $\mathcal{R} = \{\mathbf{R}_1, \mathbf{R}_2 \cdots \mathbf{R}_{|\mathcal{R}|}\}$ is the set of all links (called factual links), where each set $\mathbf{R}_r : \mathbf{E}_1(r) \rightarrow \mathbf{E}_2(r)$ is a specific relation $r \in \{1, 2, \cdots |\mathcal{R}|\}$ between head and tail entity sets $\mathbf{E}_1(r), \mathbf{E}_2(r) \in \mathcal{E}$. Each factual link $(e_1, r, e_2) \in \mathbf{R}_r$ denotes head and tail entities $e_1 \in \mathbf{E}_1(r)$, $e_2 \in \mathbf{E}_2(r)$ with relation r. $\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2$ denote the d-dimensional entity embeddings of e_1 and e_2. For each relation r, we also learn d-dimensional head and tail embedding projectors $(\vec{\mathbf{p}}_1(r), \vec{\mathbf{p}}_2(r))$.

Task-Model Notations: Task-Model $\mathcal{M}(r)$ predicts relation-r links between entity sets $\mathbf{E}_1(r)$ and $\mathbf{E}_2(r)$. Each $\mathcal{M}(r)$ is trained with factual links $\mathbf{R}_r$.

Model-Biased Links: We predict new links (e'_1, r, e'_2) via task-model $\mathcal{M}(r)$ between the input entity $e'_1 \in \mathbf{E}_1(r)$ and the model predicted output $e'_2 \in \mathbf{E}_2(r)$ (e.g., a specific user e'_1 and a specific book e'_2 from the recommender task-model in Fig. 1). Note that factual links $(e_1, r, e_2) \in \mathbf{R}_r$ exist apriori in the knowledge graph unlike model-biased links (denoted $(e'_1, r, e'_2) \in \mathbf{R'}_r$).

3 Knowledge Graph Embeddings

Knowledge graph embedding techniques typically encode static node connectivity pattens to mitigate link-sparsity (Sun et al. (2019)) such as:

- **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)
- **Inversion:** $(e_1, r_a, e_2) \implies (e_2, r_b, e_1)$
- **Composition:** (e_1, r_a, e_2) *and* $(e_2, r_b, e_3) \implies (e_1, r_c, e_3)$
- **Analogy:** (e_1, r_a, e_2) *and* $(e_3, r_a, e_4) \implies (e_1, r_b, e_3)/(e_2, r_c, e_4)$

None of these first-cut patterns are task-specific. Prior approaches in this vein do not provide mechanisms for task-adaptation or multi-task learning. We formalize task-to-task knowledge transfer as follows:

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

Note that the solution to the above transfer learning is specific to the relation types r_a, r', r'' as well the entity nodes e_1 and e_2, and thus can be combined with task-models $\mathcal{M}(r)$ involving these entities or relations. We thus propose a two-step solution where we first leverage the static patterns to generate first-cut embeddings and then augment them with task-specific residual functions (Sect. 3.2) to enable adaptation to the respective task-models.

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

3.1 Link Embedding Model

Parallelizable embedding learning is critical for knowledge graph applications owing to their massive sizes. DistMult (Yang et al. (2014)) describes a block-optimizable bilinear form with a learnable diagonal embedding projector ($\mathbf{P}_r$) for each relation type r Lerer et al. (2019). Under this approach, the likelihood of a link (e_1, r, e_2) is given by:

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

However, due to the symmetric nature of the above transformation, it cannot encode anti-symmetry and inversion patterns (Sun et al. (2019)). In contrast, other methods that do not have a symmetric objective wrt. head and tail entities (e.g., Sun et al. (2019)) pose block optimization constraints. To overcome these limitations, we break the symmetry in Eq. (1) by describing two projectors (for the head and tail entity embeddings) for each relation type. Our form adds twice as many relation-specific projectors. However, the number of relation-types is typically orders of magnitude less than the number of nodes so that the overhead is insignificant. We now define the likelihood of a link (e_1, r, e_2):

$$\mathcal{L}(\vec{\mathbf{e}}_1, r, \vec{\mathbf{e}}_2) = \textit{cosine-sim}\Big(\vec{\mathbf{e}}_1 \otimes \vec{\mathbf{p}}_1(r),\ \ \vec{\mathbf{e}}_2 \otimes \vec{\mathbf{p}}_2(r)\Big) \tag{2}$$

The above modification enables composition, inversion, and anti-symmetry:

- **Anti-Symmetry:** Consider relations r_a to be anti-symmetric, so that, $(e_1, r_a, e_2) \implies \textit{not}\ (e_2, r_a, e_1)$ We can encode this in our likelihood term with orthogonal projectors for the head and tail, i.e., $\vec{\mathbf{p}}_1(r) \perp \vec{\mathbf{p}}_2(r)$ so that we take the orthogonal 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:** Relation r_c composes r_a and r_b if $(e_1, r_a, e_2), (e_2, r_b, e_3) \implies (e_1, r_c, e_3)$. 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 scale factor to Equation (2) (*sim* = *cosine-similarity*):

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

In the next subsection, we describe task-specific embedding adaptation and link-sparsity mitigation on the first-cut factual embeddings from Eq. (3).

3.2 Embedding Augmentation via Model-Biased Links

Consider the prediction task for relation r between entity sets $\mathbf{E}_1(r)$, $\mathbf{E}_2(r)$. Task-model $\mathcal{M}(r)$ predicts model-biased links $(e'_1(r), r, e'_2(r))$ where $e'_1(r) \in \mathbf{E}_1(r)$, $e'_2(r) \in \mathbf{E}_2(r)$, from its inferred co-occurrence distribution. In this manner, each $\mathcal{M}(r)$ generates model-biased links $\mathbf{R}'_r$ different from the factual links $\mathbf{R}_r$ of the same relation type. Under Eq. (3), the likelihood of each factual link $(e_1, r, e_2) \in \mathbf{R}_r$ is given by:

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

Upon optimization, we obtain the first-cut factual embedding space $\vec{\mathbf{E}}$ with the latent factual embedding distribution $P(\vec{\mathbf{E}})$. However, each task-model $\mathcal{M}(r)$ represents a co-occurrence distribution between entity sets $\mathbf{E}_1(r), \mathbf{E}_2(r)$ which differs from those in $P(\vec{\mathbf{E}})$, depending on the specific task and the model-architecture (inductive bias). We thus learn model-specific embedding spaces $\vec{\mathbf{E}}'_r$ by optimizing Eq. (3) over the model-biased links $\mathbf{R}'_r$ instead of $\mathbf{R}_r$ (Fig. 2).

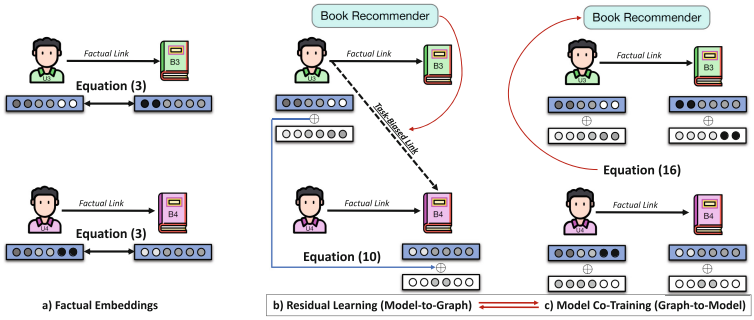

Fig. 2. (a) We learn the facutal entity embeddings via Eq. (3), (b) we then generate model-biased links with the *Book Recommender* model to train residual functions (Eq. (7)), (c) improve the task-model with the residual functions from step (b) in Eq. (12). Steps (b), (c) can be iteratively optimized.

Thus for pairs of entities $e_1 \in \mathbf{E}_1(r)$, $e_2 \in \mathbf{E}_2(r)$, we obtain both factual and model-biased embeddings ($\vec{\mathbf{e}'}$ denotes the model-biased embedding of entity e):

$$\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2 \sim P(\vec{\mathbf{E}}); \quad \vec{\mathbf{e}'}_1, \vec{\mathbf{e}'}_2 \sim P(\vec{\mathbf{E}}'_r) \tag{5}$$

We learn the divergence $\boldsymbol{\Delta}(r)$ between distributions $P(\vec{\mathbf{E}})$ and each $P(\vec{\mathbf{E}}'_r)$ so that the knowledge graph embeddings can be adapted to each task-model:

$$\boldsymbol{\Delta}(r) = \mathbf{KL}(P(\vec{\mathbf{E}}),\ P(\vec{\mathbf{E}}'_r)) \tag{6}$$

We encode $\boldsymbol{\Delta}(r)$ for each task-model $\mathcal{M}(r)$ via embedding residual shifts motivated by covariate domain-shift theory (He et al. (2016), Johansson et al. (2016)). In the next subsection, we show how this enables task→graph and graph→task embedding conversion via task-specific residual functions.

3.3 Residual Shift

The factual and model-biased embedding distributions ($P(\vec{\mathbf{E}})$, $P(\vec{\mathbf{E}}'_r)$) represent different covariate-shifts in the node embedding space depending on the biases of each task-model $\mathcal{M}(r)$. We model each of these shifts with a task-specific residual function $\boldsymbol{\delta}_r$ to translate between the spaces $\vec{\mathbf{E}}$ and $\vec{\mathbf{E}}'_r$):

$$\vec{\mathbf{e}}'_1 \;=\; \vec{\mathbf{e}}_1 \;+\; \boldsymbol{\delta}_r\,(\vec{\mathbf{e}}_1); \quad \vec{\mathbf{e}}'_2 \;=\; \vec{\mathbf{e}}_2 \;+\; \boldsymbol{\delta}_r\,(\vec{\mathbf{e}}_2) \tag{7}$$

where $\vec{\mathbf{e}}_1$ denotes the factual embedding of the entity $e_1(r)$ and each residual function $\boldsymbol{\delta}_r$ is given by,

$$\boldsymbol{\delta}_r\,(\vec{\mathbf{e}}) = tanh(\;\boldsymbol{W}_r\;(\vec{\mathbf{e}}) + \mathbf{b}_r\;) \tag{8}$$

We learn the weights $\mathbf{W}_r$ and biases $\mathbf{b}_r$ to optimize the likelihoods of the model-biased links $(\mathcal{L}(\vec{\mathbf{e}}'_1, r, \vec{\mathbf{e}}'_2)\;\forall\;(e'_1, r, e'_2) \in \mathbf{R}'_r)$ by placing the residual shifted entity embeddings $\vec{\mathbf{e}}'_1$, $\vec{\mathbf{e}}'_2$ in Eq. (3).

4 Training Methods

4.1 Learning the Task-Specific Residual Functions

We generate the model-biased links $(e'_1, r, e'_2) \in \mathbf{R}'_r$ for each $e'_1 \in \mathbf{E}_1(r)$ via $\mathcal{M}(r)$. We then learn the residual function $\boldsymbol{\delta}_r$ via alternating optimization of the following likelihoods:

$$\mathcal{L}(\mathbf{R}_r) = \sum_{(e_1,r,e_2)\in\mathbf{R}_r} \log \mathcal{L}\,(\,\vec{\mathbf{e}}_1,\, r,\, \vec{\mathbf{e}}_2\,) \tag{9}$$

$$\mathcal{L}(\mathbf{R}'_r) = \sum_{(e'_1,r,e'_2)\in\mathbf{R}'_r} \log \mathcal{L}\,(\,\vec{\mathbf{e}}'_1,\, r,\, \vec{\mathbf{e}}'_2\,) \tag{10}$$

with notations following from Eq. (3), Eq. (7) and Table 1.

4.2 Graph and Model Co-training

We now describe our training approach to concurrently learn entity embeddings and task-models with continuous differentiable objective functions. In Eq. (10), the task-model is held constant, i.e., we only learn the entity embeddings and residual functions. For co-training, we apply the same residual transformations to the factual links in the graph; and add them to the task-model's optimization objective as soft-criteria.

Table 1. Residual function notations

Symbol	Description
$\vec{\mathbf{e}}_1$, $\vec{\mathbf{e}}_2$	Factual embeddings
$\boldsymbol{\delta}_r\,(.)$	Residual function for $\mathcal{M}(r)$
$\boldsymbol{W}_r$, $\mathbf{b}_r$	Weight, bias for $\boldsymbol{\delta}_r$
$\vec{\mathbf{e}}'_1$, $\vec{\mathbf{e}}'_2$	Residual shifted embeddings
	$\vec{\mathbf{e}}'_1 \;=\; \vec{\mathbf{e}}_1 \;+\; \boldsymbol{\delta}_r\,(\vec{\mathbf{e}}_1)$
	$\vec{\mathbf{e}}'_2 \;=\; \vec{\mathbf{e}}_2 \;+\; \boldsymbol{\delta}_r\,(\vec{\mathbf{e}}_2)$

For each factual link $(e_1, r, e_2) \in \mathbf{R}_r$, we estimate the residual shifted likelihood as follows:

$$\mathcal{SA}(e_1, e_2) = \mathcal{L}(\vec{\mathbf{e}'}_1, r, \vec{\mathbf{e}'}_2) \tag{11}$$

where $\mathcal{L}$ follows from Eq. (3). We now add the following regularization term to the objective function $\mathcal{O}(r)$ of $\mathcal{M}(r)$:

$$\boldsymbol{\lambda}(r)\Big(\sum_{\mathbf{R}_r} \mathcal{SA}\,(e_1, e_2) - \mathcal{M}(r)\,(e_1, e_2)\Big) \tag{12}$$

Here, $\mathcal{M}(r)\,(e_1, e_2)$ indicates the confidence score assigned by $\mathcal{M}(r)$ to link e_2 to e_1 and $\lambda(r)$ is the regularization strength.

4.3 Model to Model Cross-Training

Let us consider the following direction of transfer, $\mathcal{M}(r_1) \rightarrow \mathcal{M}(r_2)$ (teacher-model $\rightarrow$ student-model). To cross-train $\mathcal{M}(r_2)$ with $\mathcal{M}(r_1)$, we need at least one entity set to be shared across the two models. Let us denote a shared entity set $\mathbf{E}$ with factual embeddings $\vec{\mathbf{e}}, e \in \mathbf{E}$ obtained via Eq. (3). We then learn the residual function $\boldsymbol{\delta}_{r_1}$ corresponding to the teacher-model $\mathcal{M}(r_1)$, and update the entity embeddings for $\mathbf{E}$ with Eq. (10), while holding $\boldsymbol{\delta}_{r_1}$ constant. Finally, we perform the graph-to-model updates described in Sect. 4.2 to train student-model $\mathcal{M}(r_2)$ with the updated embeddings.

5 Experimental Results

Here, 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 node embedding quality 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. We also 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.

5.1 Data Description, Setup

Google Local Reviews Dataset: He et al. (2017); Pasricha and McAuley (2018): Users rate businesses on a 0–5 scale with temporal, spatial, and textual context features in each review. We filter this dataset for at least 10 users per business and 5 businesses per user recursively and eliminate all reviews with

less than a 3-star rating. The resulting dataset has 38,614 users and 26,922 businesses, and contextual node types - Review Words (5000 nodes), Business Name Words (2000 nodes), Categories of the Business (650 nodes), Pricey-ness (4 nodes), Location (312 nodes) - states, cities, and Time (23 nodes) - time (binned into 6-h chunks), month, day.

We create our knowledge graph by connecting all users to the businesses they rated, business name and review words to each business, review words, categories of visits, and business names to users who rated them, the pricey-ness, 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). These levels constitute our relation types. The total number of nodes and links in the graph is 73,525 and 7,325,614 respectively.

Yelp Challenge Dataset: Users rate businesses on a 0–5 scale with temporal, spatial, and textual context features for each review. We filter this dataset for at least 30 users per business and 10 businesses per user recursively and eliminate all reviews with less than a 3-star rating. The resulting dataset has 25,3695 users and 69,738 businesses. We obtain the following contextual nodes - Review Words (2000 nodes), Business Attributes (200 nodes), Location (1062 nodes) - states, cities, lat-long (binned using a KD-tree), Time (23 nodes) - time (binned by 6-h 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). These levels constitute our relation types. The total number of nodes and links in the graph is 99,906 and 10,102,877 respectively.

Baselines: We choose a broad array of diverse knowledge graph embedding baselines as a representative set to evaluate the edge completion task: TransE Bordes et al. (2013), DistMult Yang et al. (2014), ComplEx Trouillon et al. (2016), Rotate Sun et al. (2019), RotH Chami et al. (2020) and GAAT Wang et al. (2019b). We used the OpenKE implementations[3] in Tensorflow/PyTorch with default parameter settings, wherever applicable.

5.2 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 Mikolov et al. (2013) on the set of review text words, business names, and all the business attributes text over all the reviews in the dataset. We use the basic version (non-transfer) of the context-aware recommender proposed in Krishnan et al. (2020)

[3] http://139.129.163.161//.

Table 2. Overall Link Prediction Results. Bold-font denotes statistically significant gains over all baselines at the 0.05 significance-level under *paired t-tests*, while * denotes the second-best performer.

Link type	User to business		User to category	
Metric	R @ 5	R @ 10	R @ 5	R @ 10
TransE [Bordes et al. 13]	0.43	0.60	0.52	0.68
RotatE [Sun et al. 19]	0.59*	0.72	0.64	0.80
RotH [Chami et al. 20]	0.58	0.76*	0.65*	0.79
DistMult [Yang et al. 14]	0.56	0.70	0.63	0.77
CompleX [Trouillon et al. 15]	0.57	0.70	0.61	0.76
GAAT [Wang et al. 19]	0.59*	0.74	0.63	0.82*
MutatE-F	0.58	0.73	0.64	0.79
MutatE-CF	**0.62**	**0.80**	**0.68**	0.84

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.

Parameters: In both the above datasets, for the context-aware recommendation model Krishnan et al. (2020), 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 Eq. (10) 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.

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** metric for the respective ranked lists. Specifically, we measure the **Recall@5**, **Recall@10** for two types of held-out links - *User* $\rightarrow$ *Business* and *User* $\rightarrow$ *Category-word* (Attribute in case of yelp), for a 100-length ranked list.

5.3 Primary Results - Link Prediction

We evaluate the above two knowledge graphs on the link completion task. We randomly tag 20% of the user nodes as held-out nodes. We then held out two

[4] https://pypi.org/project/gensim/.

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. These two link types directly correspond to the two task models we used: The word2vec model predicts user-business category word links while 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, we use the top-5 words predicted by the word2vec model, and the top-5 businesses predicted by the recommender 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 held-out links using their trained embeddings.

Key Observations from Table 2**:** The relative order of performance of the baselines is as expected, DistMult Yang et al. (2014) performs moderately owing to the inverse nature of some relation-types in our graphs across user-context-business paths. In contrast, our base model can overcome this challenge and perform 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. The two external task models, namely word2vec and the context-aware recommender, can better predict the missing links and enrich the graph compared to the heuristic or path-based link completion approach in the other baselines. It is easy to see how we can leverage the inductive biases of the specific models. While the word2vec model can interpret the review text's distributional properties, the context-aware recommender leverages the multiplicative predictors from the context features. Also, note that these two models use the same data as the Knowledge Graphs and do not depend on any external sources.

5.4 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 knowledge graph embeddings, as described in Eq. (9). Simultaneously, we make predictions from the updated embeddings for users and use these to augment the recommendation loss function as described in Eq. (11).

Table 3. Co-training performance gains against the information-flow parameter λ^j

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%

Although we did not achieve a dramatic performance difference, we observe that overregularizing the model or underregularizing the model is suboptimal. In other words, the co-training proceeds best when we set the regularizer λ^j to an optimal balance. The numbers in Table 3 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 trade-off between the forward and reverse information flow.

5.5 Cross-Training Across Tasks

Next, we describe our cross-training approach for the recommender model by leveraging the word2vec model.

Table 4. Cross-training performance gains for the context-recommender with word2vec, $\mathcal{M}^{word2vec} \rightarrow$ *Knowledge Graph* $\rightarrow \mathcal{M}^{context\text{-}aware\text{-}recommender}$, parameter λ^j is set to varying values as in Eq. (10), percentages relative to isolated performance

λ^j	e^{-5}	e^{-4}	e^{-3}	e^{-2}	e^{-1}
Context recommender	−1.2%	**+6.4%**	**+12.9%**	−10.3%	−22.1%

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 from Sect. 4.3. We then use the reverse direction to regularize the recommender model as in Eq. (12), i.e., knowledge now flows from the updated 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 informative of both user and business embeddings owing to their shared link structure, we were able to achieve noticeable performance gains for the recommender model (Table 4) after leveraging the sequence of steps described in Sect. 4.3.

Table 5. Cross-training performance gains for the word2vec model, $\mathcal{M}^{context\text{-}aware\text{-}recommender} \rightarrow$ *Knowledge Graph* $\rightarrow \mathcal{M}^{word2vec}$, parameter λ^j is again set to varying values as in Eq. (10), percentages relative to isolated performance

λ^j	e^{-5}	e^{-4}	e^{-3}	e^{-2}	e^{-1}
Word2vec	−7.9%	**−2.1%**	**−1.6%**	−4.1%	−18.3%

However, we observe that the reverse transfer direction, i.e. context-aware recommender to word2vec model, does not result in noticeable performance gains (Table 5), indicating the importance of choosing the more informative model to enrich the knowledge graph.

5.6 Sparsity Analysis

In this subsection, we 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 the performance of MUTATE and MUTATE-F for the different sparsity sets.

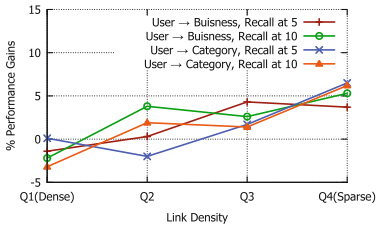

Fig. 3. The gains of MUTATE-CF relative to MUTATE-F on the two types of link prediction. In each case, we measure the performance gains across 4 quartiles of users, arranged by the density of that specific type of link for the user.

$\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$ 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 Fig. 3. As expected, we obtain the strongest gains for sparse users, i.e., users in quartiles Q3/Q4, since they lack the word-associations to help us learn better embeddings. Thus, the distributional knowledge encoded in the word2vec model bridges this gap in the knowledge graph and enriches the corresponding node embeddings.

5.7 Limitations and Discussion

The two primary limitations of our work are the non-exchangeability of cross-training and homoscedastic embedding assumption in each entity set. This results from our assumption that a single residual function, conditioned on the node embeddings, can encode the distributional differences introduced by the task-models. Alternatives such as Gaussian mixture embedding spaces (Casale et al. (2018)) can encode heteroscedastic node embeddings. However, they are quite hard to implement efficiently within a knowledge graph neural network optimization framework. We plan to study the trade-offs between generalizability and overall exchangeability in future work.

6 Conclusion

We propose a holistic view of knowledge graphs and multi-task learning to enable task-enhancement and graph enrichment. Our framework unifies co-dependent task distributions with the underlying knowledge graph via residual learning. The key strength of our approach lies in delegating the extraction of task-specific distributions to the respective task-models while enabling cross-task knowledge transfer. While the current work primarily demonstrates empirical applications of such a framework, we intend to study the theoretical exchangeability of the proposed method for future work.

References

Ai, Q., Azizi, V., Chen, X., Zhang, Y.: Learning heterogeneous knowledge base embeddings for explainable recommendation. Algorithms **11**, 9 (2018)

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems (NIPS) (2013)

Cao, Y., Wang, X., He, X., Hu, Z., Chua, T.-S.: Unifying knowledge graph learning and recommendation: towards a better understanding of user preferences. In: The World Wide Web Conference (WWW) (2019)

Casale, F.P., Dalca, A., Saglietti, L., Listgarten, J., Fusi, N.: Gaussian process prior variational autoencoders. In: Advances in Neural Information Processing Systems (2018)

Chami, I., Wolf, A., Juan, D.-C., Sala, F., Ravi, S., Ré, C.: Low-dimensional hyperbolic knowledge graph embeddings. arXiv preprint arXiv:2005.00545 (2020)

Cheng, D., Yang, F., Wang, X., Zhang, Y., Zhang, L.: Knowledge graph-based event embedding framework for financial quantitative investments. In: International Conference on Research and Development in Information Retrieval (SIGIR) (2020)

Daume III, H., Marcu, D.: Domain adaptation for statistical classifiers. J. Artif. Intell. Res. **26**, 101–126 (2006)

Ernst, P., Siu, A., Weikum, G.: KnoWlife: a versatile approach for constructing a large knowledge graph for biomedical sciences. BMC Bioinform. **16**, 1 (2015)

Guo, S., Wang, Q., Wang, L., Wang, B., Guo, L.: Knowledge graph embedding with iterative guidance from soft rules. arXiv preprint arXiv:1711.11231 (2017)

He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)

He, R., Kang, W.-C., McAuley, J.: Translation-based recommendation. In: Proceedings of the Eleventh ACM Conference on Recommender Systems (2017)

Huang, X., Zhang, J., Li, D., Li, P.: Knowledge graph embedding based question answering. In: International Conference on Web Search and Data Mining (WSDM) (2019)

Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP) (2015)

Ji, G., Liu, K., He, S., Zhao, J.: Knowledge graph completion with adaptive sparse transfer matrix. In: Schuurmans, D., Wellman, M.P. (eds.) International Conference on Artificial Intelligence (AAAI) (2016)

Jia, Y., Wang, Y., Lin, H., Jin, X., Cheng, X.: Locally adaptive translation for knowledge graph embedding. In: International Conference on Artificial Intelligence (AAAI) (2016)

Johansson, F., Shalit, U., Sontag, D.: Learning representations for counterfactual inference. In: International Conference on Machine Learning (ICML) (2016)

Krishnan, A., Das, M., Bendre, M., Yang, H., Sundaram, H.: Transfer learning via contextual invariants for one-to-many cross-domain recommendation. arXiv preprint arXiv:2005.10473 (2020)

Krishnan, A., Sharma, A., Sundaram, H.: 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) (2018)

Lerer, A., et al.: PyTorch-BigGraph: a large-scale graph embedding system. arXiv preprint arXiv:1903.12287 (2019)
Li, W.: Zipf's law everywhere. Glottometrics **5**, 14–21 (2002)
Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: International Conference on Artificial Intelligence (AAAI) (2015)
Mansour, Y., Mohri, M., Rostamizadeh, A.: Domain adaptation: learning bounds and algorithms. arXiv preprint arXiv:0902.3430 (2009)
Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems (NIPS) (2013)
Vu, T., Nguyen, T.D., Nguyen, D.Q., Phung, D.: 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) (2019)
Nickel, M., Tresp, V.: An analysis of tensor models for learning on structured data. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8189, pp. 272–287. Springer, Heidelberg (2013a). https://doi.org/10.1007/978-3-642-40991-2_18
Nickel, M., Tresp, V.: Tensor factorization for multi-relational learning. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8190, pp. 617–621. Springer, Heidelberg (2013b). https://doi.org/10.1007/978-3-642-40994-3_40
Pasricha, R., McAuley, J.: Translation-based factorization machines for sequential recommendation. In: International Conference on Recommender Systems (RecSys) (2018)
Sun, Z., Deng, Z.-H., Nie, J.-Y., Tang, J.: RotatE: knowledge graph embedding by relational rotation in complex space. arXiv preprint arXiv:1902.10197 (2019)
Sun, Z., Yang, J., Zhang, J., Bozzon, A., Huang, L.-K., Xu, C.: Recurrent knowledge graph embedding for effective recommendation. In: International Conference on Recommender Systems (RecSys) (2018)
Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International Conference on Machine Learning (ICML) (2016)
Wang, R., Li, B., Hu, S., Du, W., Zhang, M.: Knowledge graph embedding via graph attenuated attention networks. IEEE Access **8**, 5212–5224 (2019)
Wang, X., He, X., Cao, Y., Liu, M., Chua, T.-S.: KGAT: knowledge graph attention network for recommendation. In: International Conference on Knowledge Discovery & Data Mining (SIGKDD) (2019)
Wang, Z., et al.: XLore: a large-scale English-Chinese bilingual knowledge graph. In: International Semantic Web Conference (ISWC) (2013)
Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: International Conference on Artificial Intelligence (AAAI) (2014)
Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)