KNOWLEDGE ROUTER: Learning Disentangled Representations for Knowledge Graphs

Shuai Zhang[1], **Xi Rao**[1], **Yi Tay**[2] and **Ce Zhang**[1]
[1]ETH Zurich, Switzerland
[2]Google Research, USA

Abstract

The design of expressive representations of entities and relations in a knowledge graph is an important endeavor. While many of the existing approaches have primarily focused on learning from relational patterns and structural information, the intrinsic complexity of KG entities has been more or less overlooked. More concretely, we hypothesize KG entities may be more complex than we think, i.e., an entity may wear many hats and relational triplets may form due to more than a single reason. To this end, this paper proposes to learn disentangled representations of KG entities - a new method that disentangles the inner latent properties of KG entities. Our disentangled process operates at the graph level and a neighborhood mechanism is leveraged to disentangle the hidden properties of each entity. This disentangled representation learning approach is model agnostic and compatible with canonical KG embedding approaches. We conduct extensive experiments on several benchmark datasets, equipping a variety of models (DistMult, SimplE, and QuatE) with our proposed disentangling mechanism. Experimental results demonstrate that our proposed approach substantially improves performance on key metrics.

1 Introduction

Knowledge graphs (KG) have emerged as a compelling abstraction for organizing structured knowledge. They have been playing crucial roles in many machine learning tasks. A knowledge graph represents a collection of linked data, describing entities of interest and relationships between them. To incorporate KGs into other machine learning systems, a prevalent way is mapping entities and relations of knowledge graphs into expressive representations in a low-dimensional space that preserves the relationships among objects, also known as knowledge graph embeddings. Representative work such as (Bordes et al., 2013; Wang et al., 2014; Yang et al., 2014; Sun et al., 2019; Zhang et al., 2019; Chami et al., 2020) has gained intensive attention across the recent years.

The substantial effectiveness of recent work can be attributed to relational pattern modeling in which a suitable relational inductive bias is used to fit the structural information in data. Nevertheless, these methods ignore the fact that the origination and formation of KGs can be rather complex (Ehrlinger and Wöß, 2016). They may be collected, mined, handcrafted or merged in a complicated or convoluted process (Ji et al., 2017; Bosselut et al., 2019; Qin et al., 2018). To this end, entities in a knowledge graph may be highly entangled and relational triplets may form and be constructed for various reasons under a plethora of different circumstances or contexts. Contextual reasons and/or domains may be taken into account at the same time. As such, it is only natural that KG embedding methods trained in this fashion would result in highly entangled latent factors. Moreover, the existing holistic approaches fail to disentangle such factors and may result in sub-optimal solutions.

Recently, disentangled representation learning has achieved state-of-the-art performance and attracts much attention in the field of visual representation learning. A disentangled representation should separate the distinct, informative factors of variations in the data (Bengio et al., 2013). Disentangling the latent factors hidden in the observed data can not only increase the robustness, making the model less sensitive to misleading correlations but also enhance the model explainability. Disentanglement can be achieved using either supervised signals or unsupervised approaches. Zhu et al. (Zhu et al., 2014) propose to untangle the identity and view features in a supervised face recognition task. A bilinear model is adopted in (Tenenbaum and Freeman, 2000) to separate content from styles. There is also a large body of work on unsupervised disentangled representation learning (Chen et al.,

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1–10
June 6–11, 2021.

2016; Denton et al., 2017; Higgins et al., 2016). Generally, the disentanglement mechanism is integrated into unsupervised learning frameworks such as variational autoencoders (Kingma and Welling, 2013) and generative adversarial networks (Goodfellow et al., 2014). The quality of unsupervised disentangled representation can even match that learned from supervised label signals.

Inspired by the success of disentangled representation learning, we seek to enhance the disentanglement capability of entities representation in knowledge graphs. Our hope is that this idea can address the aforementioned challenge in learning entity embeddings, that is, enabling the entities embeddings to better reflect the their inner properties. Unlike learning disentangled representations in visual data, it is more challenging to disentangle the discrete relational data. Most KGs embedding approaches operate at the triplet level, which is uninformative for disentanglement. Intuitively, information about the entities resides largely within the graph encoded through neighborhood structures. Our assumption is that an entity connects with a certain group of entities for a certain reason. For example, Tim Robbins, as an actor, starred in films such as *The Shawshank Redemption*; as a musician, is a member of the folk music group *The Highwaymen*. We believe that relational triplets form because of different factors and this can be disentangled when looking it at the graph level.

To summarize, our key contributions are: (1) We propose **Knowledge Router** (KR), an approach that learns disentangled representations for entities in knowledge graphs. Specifically, a neighbourhood routing mechanism disentangles the hidden factors of entities from interactions with their neighbors. (2) Knowledge Router is model agnostic, which means that it can play with different canonical knowledge graph embedding approaches. It enables those models to have the capability in learning disentangled entity representations without incurring additional free parameters. (3) We conduct extensive experiments on four publicly available datasets to demonstrate the effectiveness of Knowledge Router. We apply Knowledge Router to models such as DistMult, SimplE, and QuatE and observe a notable performance enhancement. We also conduct model analysis to inspect the inner workings of Knowledge Router.

2 Related Work

2.1 Learning Disentangled Representations

Learning representations from data is the key challenge in many machine learning tasks. The primary posit of disentangled representation learning is that disentangling the underlying structure of data into disjoint parts could bring advantages.

Recently, there is a growing interest in learning disentangled representations across various applications. A trending line of work is integrating disentanglement into generative models. (Tran et al., 2017) propose a disentangled generative adversarial network for face recognition and synthesis. The learned representation is explicitly disentangled from a pose variation to make it pose-invariant, which is critical for face recognition/synthesis task. (Denton et al., 2017) present a disentangled representation learning approach for videos. The proposed approach separates each frame into a time-independent component and a temporal dynamics aware component. As such, it can reflect both the time-invariant and temporal features of a video. (Ma et al., 2018) propose a disentangled generative model for personal image generation. It separates out the foreground, background, and pose information, and offers a mechanism to manipulate these three components as well as control the generated images. Some works (Higgins et al., 2016; Burgess et al., 2018) (e.g., β-VAE) integrate disentanglement mechanism with variational autoencoder, a probabilistic generative model. β-VAE uses a regularization coefficient β to constrain the capacity of the latent information channel. This simple modification enables latent representations to be more factorised.

Drawing inspiration from the vision community, learning disentangled representations has also been investigated in areas such as natural language processing and graph analysis. (Jain et al., 2018) propose an autoencoders architecture to disentangle the populations, interventions, and outcomes in biomedical texts. (Liu et al., 2019) propose a prism module for semantic disentanglement in named entity recognition. The prism module can be easily trained with downstream tasks to enhance performance. For graph analysis, (Ma et al., 2019a) propose to untangle the node representation of graph-structured data in graph neural networks. (Ma et al., 2019b) present a disentangled variational autoencoder to disentangle the user's diverse interests for recommender systems.

2.2 Knowledge Graph Embeddings

Learning effective representations for knowledge graphs is extensively studied because of its importance in downstream tasks such as knowledge graph completion, natural language understanding, web search, and recommender systems. Among the large body of related literature, two popular lines are translational approaches and semantic matching approaches. The groundbreaking TransE (Bordes et al., 2013) sets the fundamental paradigm for translational models. Typically, the aim is to reduce the distance between translated (by relation) head entity and tail entity. Successors such as TransH (Wang et al., 2014), TransR (Lin et al., 2015) all follow this translational pattern. Semantic matching methods calculate the semantic similarities between entities. A representative semantic model is DistMult (Yang et al., 2014) which measures the plausibility of triplets with vector multiplications. To model more complex relation patterns, (Trouillon et al., 2016; Zhang et al., 2019; Sun et al., 2019; Zhang et al., 2021) extend the embedding spaces to complex number space or hyperbolic space. A fully expressive model named SimplE (Kazemi and Poole, 2018) could achieve the same level of capability of ComplEx (Trouillon et al., 2016) with lower calculation cost.

Inspired by the success of disentangled representations, we explore methods to factorize different components/aspects of entangled entities in a knowledge graph. To the best of our knowledge, our work is one of the first efforts to induce disentangled representations in knowledge graphs. Our disentangled embedding algorithm can be easily integrated into existing knowledge graph embedding models (model agnostic).

3 The Proposed Knowledge Router

3.1 Notation and Problem Formulation

Suppose we have an entity set $\mathcal{E}$ and a relation set $\mathcal{R}$, where $|\mathcal{E}| = N$ and $|\mathcal{R}| = M$. A knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ is made up of a collection of facts $\mathcal{F}$ in triplet form (h, r, t), where $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. The triplet $(h, r, t) \in \mathcal{F}$ means that entities h and r are connected via a relation r. The facts are usually directional, which means exchanging the head entity and tail entity does not necessarily result in a legitimate fact.

We are concerned with the link prediction task. The goal is to embed the entities and relations of a knowledge graph into low-dimensional representations that can preserve the facts in the graph. A classical setting is using an embedding matrix $\mathbf{E} \in \mathbb{R}^{N \times d}$ to represent all the entities and an embedding matrix $\mathbf{W} \in \mathbb{R}^{M \times d}$ to represent all the relations.

Notation	Description
$\mathcal{E}$	Entity set.
$\mathcal{R}$	Relation set.
$\mathbf{E}$	The entity embedding matrix.
$\mathbf{W}$	The relation embedding matrix.
$\mathbf{E}_e$	The e^{th} row of the entity embedding matrix.
$\mathbf{W}_r$	The r^{th} row of the relation embedding matrix.
d	The length of the embedding vector.
$\mathcal{N}(e)$	Neighbourhood entities set of entity e.
K	The number of independent components.
T	The number of routing iterations.
$\mathbf{x}_{e,k}$	The k^{th} initial vector for entity e.
$\mathbf{p}_{e,k}$	The k^{th} vector of entity e after disentanglement.
$\mathrm{s}_{e,i,k}$	The similarity score between entity e and entity i w.r.t the k^{th} component.
$\mathrm{w}_{i,k}$	The extent to which the model attends to the k^{th} component of entity i.

Table 1: The notations and denotations.

3.2 Disentangled Knowledge Graph Embeddings

Instead of directly modeling triplet facts, we propose to disentangle the entities with their neighbors in a message passing setting. The neighborhood entities could form several clusters for different reasons and the entity is updated by the information accepted from its neighborhood clusters.

Figure 1 illustrates the overall process of Knowledge Router. It consists of two stages: (1) disentangling the entities from a graph perspective using neighbourhood routing; (2) scoring the facts using relations and the disentangled entities representations.

Let us build an undirected graph from the training data. The relations are anonymized, which means we do not need to know under which conditions two entities are linked. We denote the neighbourhood of entity e as $\mathcal{N}(e)$, regardless of the relations. Our neighborhood routing approach operates on this graph.

Given an entity e, we aim to learn a disentangled embedding that encodes various attributes of the entity. In this regard, we suppose that each entity is composed of K independent components, with each component denoted by $\mathbf{p}_{e,k} \in \mathbb{R}^{\frac{d}{K}}$, where $\forall k = 1, 2, ..., K$. Each component stands for one aspect of the entity, e.g., a role of a person. A

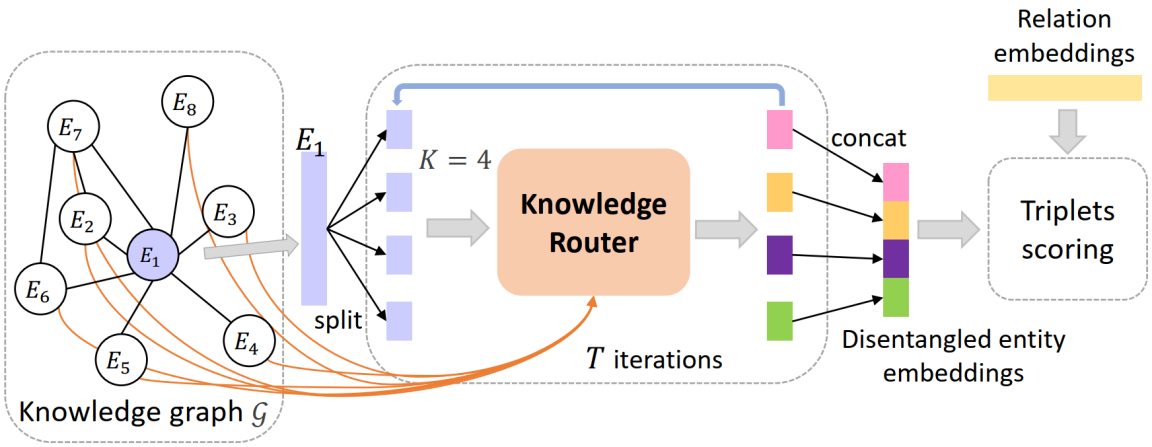


Figure 1: The overall procedure of the proposed Knowledge Router algorithm for learning disentangled entity representations. In this example, we disentangle the entity embedding into four components ($K = 4$) via neighborhood routing (iterate T times). These components are then concatenated to represent the corresponding entity.

major challenge here is to make the learned K components to be independent of one another so that different facets can be separately encoded. To this end, we adopt routing mechanisms that are inspired by capsule networks (Hinton et al., 2011). Specifically, we aim to learn the K components from both the entity e and its neighbourhoods $\mathcal{N}(e)$. Next, we describe this procedure in detail.

For each entity e, we first initialize the $\mathbf{E}_e$ randomly and evenly split it into K parts. The k^{th} part is denoted by $\mathbf{x}_{e,k} \in \mathbb{R}^{\frac{d}{K}}$. By doing so, the embedding is projected into different subspaces. To ensure computation stability, each part is also normalized as follows:

$$\mathbf{x}_{e,k} = \frac{\mathbf{x}_{e,k}}{\| \mathbf{x}_{e,k} \|_2} \tag{1}$$

This is used for the initialization of $\mathbf{p}_{e,k}$. Obviously, the information contained is limited and it cannot reach the goal of disentanglement. To enrich the information, we use a graph message passing mechanism and define the update rule for the k^{th} component of $\mathbf{p}_e$ as follows:

$$\mathbf{p}_{e,k} = \mathbf{x}_{e,k} + \text{AGGREGATE}(\{\mathbf{x}_{i,k}, \forall i \in \mathcal{N}(e)\}), \tag{2}$$

where AGGREGATE represents the neighborhood aggregation function (defined in equation 5). The same ℓ_2 normalization as (1) is applied to $\mathbf{p}_{e,k}$ afterwards.

In this way, $\mathbf{p}_{e,k}$ contains information from the k^{th} aspect of both entity e and all of its neighbors. Common aggregating functions such as mean pooling and sum pooling are viable, but treating each neighbor equally when determining one component of the representation is undoubtedly not sensible. As such, an attention mechanism is used to obtain weights for each neighbor. In particular, a scaled dot-product attention method is applied. We first get the dot product between $\mathbf{p}_{e,k}$ and $\mathbf{x}_{i,k}, \forall i \in \mathcal{N}(e)$. For each k, we get the following similarity score:

$$s_{e,i,k} = \frac{\mathbf{p}_{e,k}^{\top} \mathbf{x}_{i,k}}{\sqrt{d/k}}, \tag{3}$$

which provides information on how entity e interacts with its neighbour entity i pertaining to the aspect k. Then the softmax function is applied to get the weight distribution over different components for each neighbour.

$$w_{i,k} = \frac{\exp(s_{e,i,k})}{\sum_{k=1}^{K} \exp(s_{e,i,k})}, \tag{4}$$

and $w_{i,k}$ indicates the extent to which the model attends to the k^{th} component of entity i.

Now, we formulate the definition of the AGGREGATE function as follows:

$$\text{AGGREGATE}(\{\mathbf{x}_{i,k}, \forall i \in \mathcal{N}(e)\}) := \sum_{i \in \mathcal{N}(e)} w_{i,k} \mathbf{x}_{i,k} \tag{5}$$

The above process, including equations (2), (3), (4), (5) for learning $\mathbf{p}_{e,k}, \forall k = 1, 2, ..., K$, is repeated for T iterations, which is the same as that of a routing mechanism. Like capsule networks (Sabour et al., 2017), we also assume that entity (object) is composed of entity (object) parts. This routing method enables it to model part-whole

relationships and enlarge the differences between parts after several routing iterations.

Afterwards, the concatenation of all K components of an entity is used to represent that entity. That is, the disentangled representation $\mathbf{p}_e$ of the entity e is defined as:

$$\mathbf{p}_e = [\mathbf{p}_{e,1}, \mathbf{p}_{e,2}, ..., \mathbf{p}_{e,K}] \tag{6}$$

This neighborhood routing algorithm is model agnostic as our aim is to learn an entity embedding matrix which is necessary for most knowledge graph embedding methods. It is worth noting that this model will not introduce additional free parameters to the model.

The intuition behind the "routing mechanism" is that each facet in an entity has a separate route to contribute to the meaning of this entity. The routing algorithm will coordinately infer $\mathbf{p}_{e,k}$ (we can view it as the center of each cluster) and $w_{i,k}$ (the probability that factor k is the reason why entity e is connected with entity i). They are coordinately learned and under the constraint that each neighbor should belong to one cluster. It is reminiscent of the iterative method used in the EM algorithm (Bishop, 2006) and is expected to lead to convergence and meaningful disentangled representations (Ma et al., 2019a).

Until now, the relation embeddings are not utilized as all relations are anonymous during graph construction. This algorithm will be jointly trained with the following facts scoring algorithms.

3.3 Facts Scoring using Disentangled Entities

Using disentangled entity embeddings alone cannot recover the facts in a knowledge graph. It shall be further updated simultaneously with the relation embeddings for the fact scoring process. To predict whether a triplet $\langle h, r, t\rangle$ holds or not, we first fetch the learned disentangled representation of the head and tail entities, $\mathbf{p}_h$ and $\mathbf{p}_t$. Then we adopt three methods for triplet scoring including DistMult (Yang et al., 2014), SimplE (Kazemi and Poole, 2018), and QuatE (Zhang et al., 2019). We denote the model after disentanglement as: KR-DistMult, KR-SimplE, and KR-QuatE.

The scoring function of KR-DistMult is defined as follows:

$$\phi(h, r, t) = \langle \mathbf{W}_r, \mathbf{p}_h, \mathbf{p}_t\rangle \tag{7}$$

where $\langle *, *, *\rangle$ denotes the standard component-wise multi-linear dot product.

SimplE needs an additional entity embedding matrix $\mathbf{H} \in \mathbb{R}^{N\times d}$ and an additional relation embedding matrix $\mathbf{V} \in \mathbb{R}^{M\times d}$. We perform the same disentanglement process on $\mathbf{H}$ and denote the disentangled representation of entity e as $\mathbf{q}_e$, the scoring function of KR-SimplE (SimplE-avg is adopted since it outperforms SimplE-ignr) is:

$$\phi(h, r, t) = (\langle \mathbf{W}_r, \mathbf{p}_h, \mathbf{q}_t\rangle + \langle \mathbf{V}_r, \mathbf{q}_h, \mathbf{p}_t\rangle) \times \frac{1}{2} \tag{8}$$

For QuatE, entities and relations are represented with quaternions. Each quaternion is composed of a real component and three imaginary components. Let $\mathbf{Q} \in \mathbb{H}^{N\times d}$ denote the quaternion entity embedding and $\mathbf{W} \in \mathbb{H}^{M\times d}$ denote the quaternion relation embedding, where $\mathbb{H}$ is the quaternion space. Each entity is represented by $\mathbf{Q}_e$. We apply the Knowledge Router algorithm on each component of $\mathbf{Q}_e$. The scoring function of KR-QuatE is:

$$\phi(h, r, t) = \mathbf{Q}_h^{\text{KR}} \otimes \frac{\mathbf{W}_r}{|\mathbf{W}_r|} \cdot \mathbf{Q}_t^{\text{KR}} \tag{9}$$

where "$\otimes$" is Hamilton product; "$\cdot$" represents the quaternion inner product; $\mathbf{Q}^{\text{KR}}$ denotes the entity representation after disentanglement.

As Knowledge Router is model agnostic, other scoring functions are also applicable.

3.4 Objective Functions

To learn a disentangled KG model, we adopt the following negative log-likelihood loss:

$$\mathcal{L} = -\frac{1}{\mathcal{S}}\sum_{i=1}^{\mathcal{S}}(y^{(i)}\log(\phi^{(i)}) + (1-y^{(i)})\log(1-\phi^{(i)})) \tag{10}$$

where $\mathcal{S}$ is the number of training samples (triplets); $y^{(i)}$ is a binary label indicating whether the i^{th} triplet holds or not; $\phi^{(i)}$ is the prediction for the i^{th} triplet. Our model can be trained with commonly used minibatch gradient descent optimizers.

3.5 Complexity Analysis

The disentanglement process of each node needs $\mathcal{O}(|\mathcal{N}(e)|\frac{d}{K}K + T(|\mathcal{N}(e)|\frac{d}{K}K + \frac{d}{K}K))$ time complexity, where $|\mathcal{N}(e)|$ is neighborhood size. After simplification, the time complexity is $\mathcal{O}(T|\mathcal{N}(e)|d)$. This will not incur a high computational cost since T is usually a small number (e.g., 3), and the neighborhood size is determined by the average degree and can usually be constrainted by a constant value (e.g., 10). With regard to fact

Datasets	N	M	|train|	|validation|	|test|
FB15k-237	14,541	237	272,115	17,535	20,466
WIKIDATA	11,153	96	53,252	11,894	11,752
ICEWS14	7,128	230	42,690	7,331	7,419
ICEWS05-15	10,488	251	368,962	46,275	46,092

Table 2: Statistics of datasets used in our experiments.

scoring, it requires $\mathcal{O}(d)$ time complexity for each triplet in general.

4 Experiments

In this section, we conduct experiments on several benchmark datasets to verify the effectiveness of the proposed approach. We target at answering: **RQ I**: whether the disentanglement method can enhance the traditional knowledge graph embedding methods? **RQ II**: Model-agnosticism: can it effectively work with different baseline models? **RQ III**: How do certain important hyper-parameters impact the model performance and what has the disentanglement algorithm learned? Are they meaningful?

4.1 Datasets Description

We use four publicly available datasets including ICEWS14, ICEWS05-15, WikiData, and FB15k-237. The reason for using these is that their entities are complicated and highly entangled. The WordNet dataset is not appropriate to evaluate the proposed method as the entities in WordNet are already disentangled[1].

FB15k-237 is a subset of the Freebase knowledge base which contains general information about the world. We adopt the widely used version generated by (Dettmers et al., 2018) where inverse relations are eliminated to avoid data leakage.

WikiData is sampled from Wikidata[2], a collaborative open knowledge base. The knowledge is relatively up-to-date compared with FB15k-237. We use the version provided by (García-Durán et al., 2018). Timestamp is discarded.

ICEWS (García-Durán et al., 2018) is collected from the integrated crisis early warning system[3] which was built to monitor and forecast national and internal crises. The datasets contain political events that connect entities (e.g., countries, presidents, intergovernmental organizations) to other entities via predicates (e.g., "make a visit", "sign formal agreement", etc.). ICES14 contains events in the year 2014, while the ICEWS05-15 contains events occurring between 2005 and 2015. Temporal information is not used in our experiments.

Data statistics and the train/validation/test splits are summarized in Table 2.

4.2 Evaluation Protocol

We adopt four commonly used evaluation metrics including hit rate with given cut-off (HR@1, HR@3, HR@10) and mean reciprocal rank (MRR). HR measures the percentage of true triples of the ranked list. MRR is the average of the mean rank inverse which reflects the ranking quality. Evaluation is performed under the commonly used filtered setting (Bordes et al., 2013), which is more reasonable and stable compared to the unfiltered setting.

4.3 Baselines

To demonstrate the advantage of our approach, we compare the proposed method with several representative knowledge graph embedding approaches including TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), ComplEx (Trouillon et al., 2016), SimplE (Kazemi and Poole, 2018), and QuatE (Zhang et al., 2019). For FB15k-237, the results of RotatE (Sun et al., 2019) and R-GCN (Schlichtkrull et al., 2018) are also included.

4.4 Implementation Details

We implement our model using pytorch (Paszke et al., 2019) and run it on TITAN XP GPUs. We adopt Adam optimizer to learn our model (Goodfellow et al., 2016) and the learning rate is set to 0.01 without further tuning. The embedding size d is set to 100 and the number of negative samples is fixed to 50. The batch size is selected from $\{128, 512, 1024\}$. The regularization rate is searched from $\{0.0, 0.01, 0.1, 0.2, 0.3, 0.5\}$. For the disentanglement algorithm, the number of components K is selected from $\{2, 4, 5, 10\}$ (K should be divisible by d); the number of routing iterations T is tuned amongst $\{2, 3, 4, 5, 7, 10\}$. The hyper-parameters are determined by the validation set. Each experiment runs five times and the average is reported. For convenience of implementation, the maximum neighbor sizes are: 16 (FB15K-237), 4 (WikiData), 10 (ICEWS14), 16 (ICEWS05-15). We apply zero padding to entities that have fewer neighbors.

4.5 Main Results

The test results on the four datasets are shown in Tables 3, 4 and 5. Evidently, we can make the

[1]For example, a word with five meanings is represented with five different entities in WordNet.

[2]https://www.wikidata.org/

[3]http://www.icews.com/

Models	FB15k-237			
	MRR	HR@10	HR@3	HR@1
TransE	0.294	0.465	-	-
DistMult	0.241	0.419	0.263	0.155
ComplEx	0.247	0.428	0.275	0.158
SimplE	0.229	0.379	0.252	0.153
R-GCN♡	0.249	0.417	0.264	0.151
RotatE⋆	0.297	0.480	0.328	0.205
QuatE⋄	0.311	0.495	0.342	0.221
KR-DistMult	0.275	0.450	0.302	0.190
KR-SimplE	0.273	0.438	0.298	0.190
KR-QuatE	**0.322**	**0.507**	**0.356**	**0.228**
KR-D vs. D	+14.1%	+7.4%	+14.8%	+22.6%
KR-S vs. S	+19.2%	+15.5%	+18.2%	+24.2%
KR-Q vs. Q	+3.5%	+2.4%	+4.1%	+3.2%

Table 3: Results on the FB15K-237 dataset. Best results are in bold. "D", "S", and "D" stand for DistMult, SimplE, and QuatE, respectively. "♡": results from (Schlichtkrull et al., 2018). "⋆": results from (Sun et al., 2019). For fair comparison, adversarial negative sampling is not used. "⋄": results from (Zhang et al., 2019) (without N3 regularization and type constraints).

Models	WikiData			
	MRR	HR@10	HR@3	HR@1
TransE	0.164	0.288	0.162	0.101
DistMult	0.863	0.902	0.883	0.837
ComplEx	0.850	0.895	0.871	0.821
SimplE	0.878	0.902	0.890	0.861
QuatE	0.792	0.852	0.823	0.752
KR-DistMult	0.888	0.911	0.898	0.872
KR-SimplE	0.898	0.912	0.900	0.891
KR-QuatE	**0.900**	**0.912**	**0.900**	**0.893**
KR-D vs. D	+2.9%	+1.0%	+1.7%	+4.2%
KR-S vs. S	+2.3%	+1.1%	+1.1%	+3.6%
KR-Q vs. Q	+13.6%	+7.0%	+9.4%	+18.7%

Table 4: Results on WikiData. Best results are in bold. "D", "S", and "D" stand for DistMult, SimplE, and QuatE, respectively.

following observations: (1) Models with Knowledge Router outperform the counterparts without it by a large margin, confirming the effectiveness of Knowledge Router and assuring the benefits of learning disentangled representations. This clearly answers our **RQ I**; (2) On the four datasets, we observe a consistent enhancement of Knowledge Router on both traditional embedding models such as DistMult, SimplE, as well as hypercomplex number based model QuatE. This is expected as our Knowledge Router is model agnostic (**RQ II**) and can be integrated to canonical knowledge embedding models. (3) The model KR-QuatE is usually the best performer on all datasets, indicating the generalization capability of Knowledge Router in more complex embedding spaces.

On the FB15k-237 dataset, the model KR-QuatE achieves the best performance compared to the recent translational model RotatE and the semantic matching model QuatE. Models such as DistMult and SimplE are also outperformed by KR-DistMult and KR-SimplE. In addition, it is good to note that the performance of each of the three KR-models is much higher than the graph convolutional networks based model, R-GCN. This implies that simply/naively incorporating graph structures might not lead to good performance. Knowledge Router also operates at the graph level, moreover, the neighborhood information is effectively utilized for disentanglement.

Similar trends are also observed on WikiData. Interestingly, we find that the performance differences of the three KR-models are quite small on this dataset. We hypothesize that the performance on this dataset has already been quite high, making further improvement more difficult.

Among the baselines, SimplE is the best performer. We notice that even though the pure QuatE does not show impressive performance, the Knowledge Router enhances its results and enables it to achieve the state-of-the-art performance.

On the two ICEWS datasets, disentanglement usually leads to a large performance boost. The average performance gains of Knowledge Router based models (KR-DistMult, KR-SimplE, KR-QuatE) are high, compared with the original models (DistMult, SimplE, and QuatE). We also observe that KR-QuatE outperforms other models significantly.

To conclude, our experimental evidence shows that disentangling the entities can indeed bring performance increase and the proposed Knowledge Router can effectively be integrated into different models.

4.6 Model Analysis

To answer **RQ III** and gain further insights, we empirically analyze the important ingredients of the model via qualitative analysis and visualization.

4.6.1 Visualization of similarity scores

The attention mechanism is critical to achieving the final disentanglement. To show its efficacy, we visualize four examples of attention weights $w_{i,k}$ in Figure 2. The color scale represents the strength of the attention weights. Each row represents a neighbor of the selected entity and each column represents a disentangled component. We observe a clear staggered pattern in the attention weights. For example, in the upper left figure, the neighbors

Models	ICEWS 14				ICEWS05-15			
	MRR	HR@10	HR@3	HR@1	MRR	HR@10	HR@3	HR@1
TransE⋆	0.280	0.637	-	0.094	0.294	0.663	-	0.090
DistMult⋆	0.439	0.672	-	0.323	0.456	0.691	-	0.337
SimplE	0.458	0.687	0.516	0.341	0.478	0.708	0.539	0.359
ComplEx	0.638	**0.753**	0.677	0.574	0.708	0.821	0.748	0.645
QuatE	0.656	0.733	0.673	0.615	0.723	0.817	0.754	0.671
KR-DistMult	0.544	0.740	0.608	0.439	0.611	0.789	0.662	0.519
KR-SimplE	0.588	0.753	0.642	0.498	0.639	0.803	0.689	0.553
KR-QuatE	**0.688**	**0.753**	**0.692**	**0.643**	**0.797**	**0.853**	**0.812**	**0.767**
KR-DistMult vs. DistMult	+23.9%	+10.1%	-	+11.6%	+33.9%	+14.2%	-	+54.0%
KR-SimplE vs. SimplE	+28.3%	+9.6%	+24.4%	+46.0%	+33.7%	+13.4%	+27.8%	+54.0%
KR-QuatE vs. QuatE	+4.9%	+2.7%	+2.8%	+4.6%	+10.2%	+4.4%	+7.7%	+14.3%

Table 5: Results on ICEWS14 and ICEWS05-15. Best results are in bold. "⋆": results from (García-Durán et al., 2018). Note that the embedding size is 100 for all models.

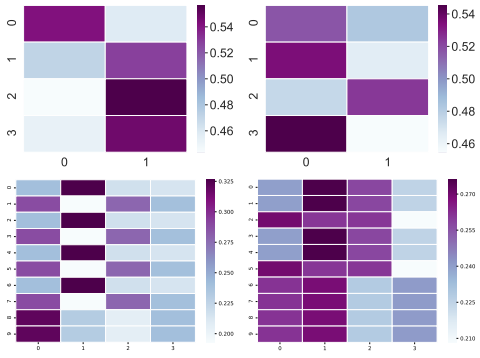
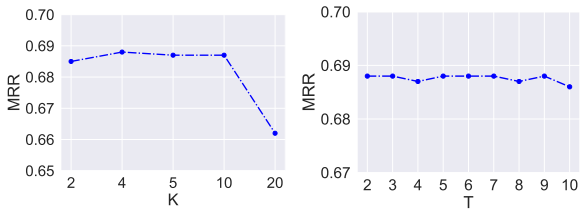


Figure 2: Four examples of attention weights learned during the routing process. Te upper two examples are taken from WikiData ($K = 2$) and the lower two examples are taken from ICEWS14 ($K = 4$). Rows represent neighbors and columns represent disentangled components. Best viewed in color.

Figure 3: Case study on WikiData for the German footballer *Michael Rensing*.



(a) (b)

Figure 4: (a) The impact of number of components K on ICEWS14. (b) The impact of number of neighborhood routing iteration T on ICEWS14.

1, 2, 3 give higher weights to the second component while 0 gives a stronger weight to the first component. In other figures, the attention weights are also staggered among the disentangled components.

4.6.2 Case study

We randomly pick one entity (*Michael Rensing*, a German footballer) from the WikiData and show the learned weight between him and his neighborhood entities in Figure 3. We observe that *FC Bayern Munich* and *Jan Kirchhoff* (who is also a team member of the *FC Bayern Munich* club) contribute more on the first component of the representation of *Michael Rensing*, while *Germany national under-18 football team* and *Germany national under-21 football team* make larger contributions to the second component. Clearly, the first component captures the fact that *Michael Rensing* is a member of the *FC Bayern Munich* association football club and the second component reflects that he is also a Germany national football team member. This case justifies our assumption that entities are connected for different reasons and demonstrates that Knowledge Router is able to disentangle the underlying factors effectively.

4.6.3 Impact of size K

We analyze the impact of K. Intuitively, K is difficult to choose since there is no prior information on how many components we should decompose each entity into. The test results with varying K on ICEWS14 of KR-QuatE are shown in Figure 4 (a).

As can be seen, using large K could result in a performance degradation. One possible reason is that there are not enough neighborhood entities to be divided into 20 groups. Empirically, we found that setting K to a small value around 2 to 5 can usually render reasonable results. A practical suggestion is that K should not exceed the average degree of the knowledge graph.

4.6.4 Impact of routing iteration T

We study the influence of number of routing iterations. As shown in Figure 4 (b), the model performance is stable when using different iterations. The reason is that the Knowledge Router algorithm is not prone to saturation and has good convergence properties. In practice, we find that using a small number of iterations (e.g., 3) could lead to ideal enhancement without putting on much computation burden.

5 Conclusion

In this paper, we present Knowledge Router, an algorithm for learning disentangled entity representations in knowledge graphs. Our method is model agnostic and can be applied to many canonical knowledge graph embedding methods. Extensive experiments on four benchmarking datasets demonstrate that equipping popular embedding models with the proposed Knowledge Router can outperform a number of recent strong baselines. Via qualitative model analysis, we discover that Knowledge Router can effectively learns the hidden factors connecting entities, thus leading to disentanglement. We also showcase the impact of certain important hyper-parameters and give suggestions on hyper-parameters tuning.

References

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

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*, pages 2787–2795.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. 2018. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. *arXiv preprint arXiv:2005.00545*.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180.

Emily L Denton et al. 2017. Unsupervised learning of disentangled representations from video. In *Advances in neural information processing systems*, pages 4414–4423.

T Dettmers, P Minervini, P Stenetorp, and S Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, volume 32, pages 1811–1818. AAI Publications.

Lisa Ehrlinger and Wolfram Wöß. 2016. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48:1–4.

Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. *arXiv preprint arXiv:1809.03202*.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. 2011. Transforming auto-encoders. In *International conference on artificial neural networks*, pages 44–51. Springer.

Sarthak Jain, Edward Banner, Jan-Willem van de Meent, Iain Marshall, and Byron C Wallace. 2018. Learning disentangled representations of texts with application to biomedical abstracts. In *Proceedings*

of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4683–4693.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in neural information processing systems*, pages 4284–4295.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2181–2187.

Kun Liu, Shen Li, Daqi Zheng, Zhengdong Lu, Sheng Gao, and Si Li. 2019. A prism module for semantic disentanglement in name entity recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5358–5362.

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019a. Disentangled graph convolutional networks. In *International Conference on Machine Learning*, pages 4212–4221.

Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019b. Learning disentangled representations for recommendation. In *Advances in Neural Information Processing Systems*, pages 5711–5722.

Liqian Ma, Qianru Sun, Stamatios Georgoulis, Luc Van Gool, Bernt Schiele, and Mario Fritz. 2018. Disentangled person image generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 99–108.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Robust distant supervision relation extraction via deep reinforcement learning. *arXiv preprint arXiv:1805.09927*.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3859–3869.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

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*.

Joshua B Tenenbaum and William T Freeman. 2000. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283.

Luan Tran, Xi Yin, and Xiaoming Liu. 2017. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1415–1424.

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

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

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*.

Shuai Zhang, Yi Tay, Wenqi Jiang, Da-cheng Juan, and Ce Zhang. 2021. Switch spaces: Learning product spaces with sparse gating. *arXiv preprint arXiv:2102.08688*.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems*, pages 2735–2745.

Zhenyao Zhu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2014. Multi-view perceptron: a deep model for learning face identity and view representations. In *Advances in Neural Information Processing Systems*, pages 217–225.