Interpretable and Compositional Relation Learning by Joint Training with an Autoencoder

Ryo Takahashi*[1] and **Ran Tian*[1]** and **Kentaro Inui[1,2]**
(* equal contribution)
[1]Tohoku University [2] RIKEN, Japan
{ryo.t, tianran, inui}@ecei.tohoku.ac.jp

Abstract

Embedding models for entities and relations are extremely useful for recovering missing facts in a knowledge base. Intuitively, a relation can be modeled by a matrix mapping entity vectors. However, relations reside on low dimension sub-manifolds in the parameter space of arbitrary matrices – for one reason, composition of two relations $\boldsymbol{M}_1, \boldsymbol{M}_2$ may match a third $\boldsymbol{M}_3$ (e.g. composition of relations `currency_of_country` and `country_of_film` usually matches `currency_of_film_budget`), which imposes compositional constraints to be satisfied by the parameters (i.e. $\boldsymbol{M}_1 \cdot \boldsymbol{M}_2 \approx \boldsymbol{M}_3$). In this paper we investigate a dimension reduction technique by training relations jointly with an autoencoder, which is expected to better capture compositional constraints. We achieve state-of-the-art on Knowledge Base Completion tasks with strongly improved Mean Rank, and show that joint training with an autoencoder leads to interpretable sparse codings of relations, helps discovering compositional constraints and benefits from compositional training. Our source code is released at `github.com/tianran/glimvec`.

1 Introduction

Broad-coverage knowledge bases (KBs) such as Freebase (Bollacker et al., 2008) and DBPedia (Auer et al., 2007) store a large amount of facts in the form of ⟨head entity, relation, tail entity⟩ triples (e.g. ⟨*The Matrix*, `country_of_film`, *Australia*⟩), which could support a wide range of reasoning and question answering applications. The Knowledge Base Completion (KBC) task aims to predict the missing part of an incomplete triple, such as ⟨*Finding Nemo*, `country_of_film`, ?⟩, by reasoning from known facts stored in the KB.

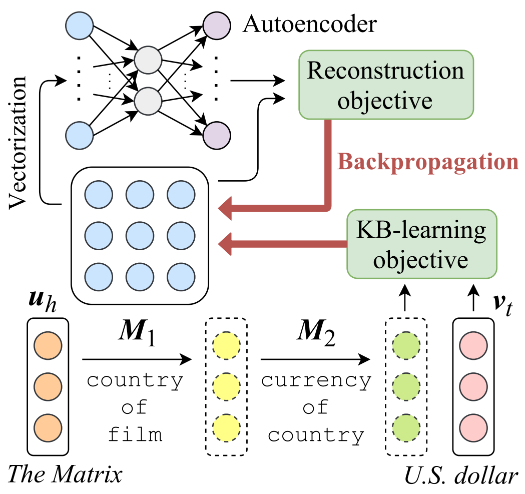

Figure 1: In joint training, relation parameters (e.g. $\boldsymbol{M}_1$) receive updates from both a *KB-learning objective*, trying to predict entities in the KB; and a *reconstruction objective* from an autoencoder, trying to recover relations from low dimension codings.

As a most common approach (Wang et al., 2017), modeling entities and relations to operate in a low dimension vector space helps KBC, for three conceivable reasons. First, when dimension is low, entities modeled as vectors are forced to share parameters, so "similar" entities which participate in many relations in common get close to each other (e.g. *Australia* close to *US*). This could imply that an entity (e.g. *US*) "type matches" a relation such as `country_of_film`. Second, relations may share parameters as well, which could transfer facts from one relation to other similar relations, for example from ⟨*x*, `award_winner`, *y*⟩ to ⟨*x*, `award_nominated`, *y*⟩. Third, spatial positions might be used to implement *composition* of relations, as relations can be regarded as mappings from head to tail entities, and the composition of two maps can match a third (e.g. the composition of currency_of_country and country_of_film matches the relation currency_of_film_budget), which could be captured by modeling composition in a space.

However, modeling relations as mappings naturally requires more parameters – a general linear map between d-dimension vectors is represented by a matrix of d^2 parameters – which are less likely to be shared, impeding transfers of facts between similar relations. Thus, it is desired to reduce dimensionality of relations; furthermore, the existence of a composition of two relations (assumed to be modeled by matrices $\boldsymbol{M}_1, \boldsymbol{M}_2$) matching a third ($\boldsymbol{M}_3$) also justifies dimension reduction, because it implies a *compositional constraint* $\boldsymbol{M}_1 \cdot \boldsymbol{M}_2 \approx \boldsymbol{M}_3$ that can be satisfied only by a lower dimension sub-manifold in the parameter space[1].

Previous approaches reduce dimensionality of relations by imposing pre-designed hard constraints on the parameter space, such as constraining that relations are translations (Bordes et al., 2013) or diagonal matrices (Yang et al., 2015), or assuming they are linear combinations of a small number of prototypes (Xie et al., 2017). However, pre-designed hard constraints do not seem to cope well with compositional constraints, because it is difficult to know *a priori* which two relations compose to which third relation, hence difficult to choose a pre-design; and compositional constraints are not always exact (e.g. the composition of currency_of_country and headquarter_location usually matches business_operation_currency but not always), so hard constraints are less suited.

In this paper, we investigate an alternative approach by training relation parameters jointly with an autoencoder (Figure 1). During training, the autoencoder tries to reconstruct relations from low dimension codings, with the reconstruction objective back-propagating to relation parameters as well. We show this novel technique promotes parameter sharing between different relations, and drives them toward low dimension manifolds (Sec.6.2). Besides, we expect the technique to cope better with compositional constraints, because it discovers low dimension manifolds posteriorly from data, and it does not impose any explicit hard constraints.

[1]It is noteworthy that similar compositional constraints apply to most modeling schemes of relations, not just matrices.

Yet, joint training with an autoencoder is not simple; one has to keep a subtle balance between gradients of the reconstruction and KB-learning objectives throughout the training process. We are not aware of any theoretical principles directly addressing this problem; but we found some important settings after extensive pre-experiments (Sec.4). We evaluate our system using standard KBC datasets, achieving state-of-the-art on several of them (Sec.6.1), with strongly improved Mean Rank. We discuss detailed settings that lead to the performance (Sec.4.1), and we show that joint training with an autoencoder indeed helps discovering compositional constraints (Sec.6.2) and benefits from compositional training (Sec.6.3).

2 Base Model

A knowledge base (KB) is a set $\mathcal{T}$ of triples of the form $\langle h, r, t\rangle$, where $h, t \in \mathcal{E}$ are entities and $r \in \mathcal{R}$ is a relation (e.g. $\langle$*The Matrix*, country_of_film, *Australia*$\rangle$). A relation r has its inverse $r^{-1} \in \mathcal{R}$ so that for every $\langle h, r, t\rangle \in \mathcal{T}$, we regard $\langle t, r^{-1}, h\rangle$ as also in the KB. Under this assumption and given $\mathcal{T}$ as training data, we consider the Knowledge Base Completion (KBC) task that predicts candidates for a missing tail entity in an incomplete $\langle h, r, ?\rangle$ triple.

Most approaches tackle this problem by training a *score function* measuring the plausibility of triples being facts. The model we implement in this work represents entities h, t as d-dimension vectors $\boldsymbol{u}_h, \boldsymbol{v}_t$ respectively, and relation r as a $d \times d$ matrix $\boldsymbol{M}_r$. If $\boldsymbol{u}_h, \boldsymbol{v}_t$ are one-hot vectors with dimension $d = |\mathcal{E}|$ corresponding to each entity, one can take $\boldsymbol{M}_r$ as the adjacency matrix of entities joined by relation r, so the set of tail entities filling into $\langle h, r, ?\rangle$ is calculated by $\boldsymbol{u}_h^\top \boldsymbol{M}_r$ (with each nonzero entry corresponds to an answer). Thus, we have $\boldsymbol{u}_h^\top \boldsymbol{M}_r \boldsymbol{v}_t > 0$ if and only if $\langle h, r, t\rangle \in \mathcal{T}$. This motivates us to use $\boldsymbol{u}_h^\top \boldsymbol{M}_r \boldsymbol{v}_t$ as a natural parameter to model plausibility of $\langle h, r, t\rangle$, even in a low dimension space with $d \ll |\mathcal{E}|$. Thus, we define the score function as

$$s(h, r, t) := \exp(\boldsymbol{u}_h^\top \boldsymbol{M}_r \boldsymbol{v}_t) \qquad (1)$$

for the basic model. This is similar to the bilinear model of Nickel et al. (2011), except that we distinguish $\boldsymbol{u}_h$ (the vector for head entities) from $\boldsymbol{v}_t$ (the vector for tail entities). It has also been proposed in Tian et al. (2016), but for modeling dependency trees rather than KBs.

More generally, we consider *composition* of relations $r_1/\dots/r_l$ to model *path*s in a KB (Guu et al., 2015), as defined by $r_1, \dots, r_l$ participating in a sequence of facts such that the head entity of each fact coincides with the tail of its previous. For example, a sequence of two facts ⟨*The Matrix*, `country_of_film`, *Australia*⟩ and ⟨*Australia*, `currency_of_country`, *Australian Dollar*⟩ form a path of composition `country_of_film/currency_of_country`, because the head of the second fact (i.e. *Australia*) coincides with the tail of the first. Using the previous $d = |\mathcal{E}|$ analogue, one can verify that composition of relations is represented by multiplication of adjacency matrices, so we accordingly define

$$s(h, r_1/\dots/r_l, t) := \exp(\boldsymbol{u}_h^\top \boldsymbol{M}_{r_1} \cdots \boldsymbol{M}_{r_l} \boldsymbol{v}_t)$$

to measure the plausibility of a path. It is explored in Guu et al. (2015) to learn a score function not only for single facts but also for paths. This *compositional training* scheme is shown to bring valuable information about the structure of the KB and may help KBC. In this work, we conduct experiments both with and without compositional training.

In order to learn parameters $\boldsymbol{u}_h, \boldsymbol{v}_t, \boldsymbol{M}_r$ of the score function, we follow Tian et al. (2016) using a Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2012) objective. For each path (or triple) $\langle h, r_1/\dots, t\rangle$ taken from the KB, we generate negative samples by replacing the tail entity t with some random noise t^*. Then, we maximize

$$\mathcal{L}_1 := \sum_{\text{path}} \ln \frac{s(h, r_1/\dots, t)}{k + s(h, r_1/\dots, t)} + \sum_{\text{noise}} \ln \frac{k}{k + s(h, r_1/\dots, t^*)}$$

as our *KB-learning objective*. Here, k is the number of noises generated for each path. When the score function is regarded as probability, $\mathcal{L}_1$ represents the log-likelihood of "$\langle h, r_1/\dots, t\rangle$ being actual path and $\langle h, r_1/\dots, t^*\rangle$ being noise". Maximizing $\mathcal{L}_1$ increases the scores of actual paths and decreases the scores of noises.

3 Joint Training with an Autoencoder

Autoencoders learn efficient codings of high-dimensional data while trying to reconstruct the original data from the coding. By joint training relation matrices with an autoencoder, we also expect it to help reducing the dimensionality of the original data (i.e. relation matrices).

Formally, we define a *vectorization* $\boldsymbol{m}_r$ for each relation matrix $\boldsymbol{M}_r$, and use it as input to the autoencoder. $\boldsymbol{m}_r$ is defined as a reshape of $\boldsymbol{M}_r$ flattened into a d^2-dimension vector, and normalized such that $\|\boldsymbol{m}_r\| = \sqrt{d}$. We define

$$\boldsymbol{c}_r := \mathrm{ReLU}(\boldsymbol{A}\boldsymbol{m}_r) \tag{2}$$

as the coding. Here $\boldsymbol{A}$ is a $c \times d^2$ matrix with $c \ll d^2$, and ReLU is the Rectified Linear Unit function (Nair and Hinton, 2010). We reconstruct the input from $\boldsymbol{c}_r$ by multiplying a $d^2 \times c$ matrix $\boldsymbol{B}$. We want $\boldsymbol{B}\boldsymbol{c}_r$ to be more similar to $\boldsymbol{m}_r$ than other relations. For this purpose, we define a similarity

$$g(r_1, r_2) := \exp(\frac{1}{\sqrt{dc}} \boldsymbol{m}_{r_1}^\top \boldsymbol{B}\boldsymbol{c}_{r_2}), \tag{3}$$

which measures the length of $\boldsymbol{B}\boldsymbol{c}_{r_2}$ projected to the direction of $\boldsymbol{m}_{r_1}$. In order to learn the parameters $\boldsymbol{A}, \boldsymbol{B}$, we adopt the Noise Contrastive Estimation scheme as in Sec.2, generate random noises r^* for each relation r and maximize

$$\mathcal{L}_2 := \sum_{r \in \mathcal{R}} \ln \frac{g(r,r)}{k + g(r,r)} + \sum_{r^* \sim \mathcal{R}} \ln \frac{k}{k + g(r, r^*)}$$

as our *reconstruction objective*. Maximizing $\mathcal{L}_2$ increases $\boldsymbol{m}_r$'s similarity with $\boldsymbol{B}\boldsymbol{c}_r$, and decreases it with $\boldsymbol{B}\boldsymbol{c}_{r^*}$.

During joint training, both $\mathcal{L}_1$ and $\mathcal{L}_2$ are simultaneously maximized, and the gradient $\nabla\mathcal{L}_2$ propagates to relation matrices as well. Since $\nabla\mathcal{L}_2$ depends on $\boldsymbol{A}$ and $\boldsymbol{B}$, and $\boldsymbol{A}, \boldsymbol{B}$ interact with all relations, they promote indirect parameter sharing between different relation matrices. In Sec.6.2, we further show that joint training drives relations toward a low dimension manifold.

4 Optimization Tricks

Joint training with an autoencoder is not simple. Relation matrices receive updates from both $\nabla\mathcal{L}_1$ and $\nabla\mathcal{L}_2$, but if they update $\nabla\mathcal{L}_1$ too much, the autoencoder has no effect; conversely, if they update $\nabla\mathcal{L}_2$ too often, all relation matrices crush into one cluster. Furthermore, an autoencoder should learn from genuine patterns of relation matrices that emerge from fitting the KB, but not the reverse – in which the autoencoder imposes arbitrary patterns to relation matrices according to random initialization. Therefore, it is not surprising that a naive optimization of $\mathcal{L}_1 + \mathcal{L}_2$ does not work.

After extensive pre-experiments, we have found some crucial settings for successful training. The most important “magic” is the scaling factor $\frac{1}{\sqrt{dc}}$ in definition of the similarity function (3), perhaps being combined with other settings as we discuss below. We have tried different factors 1, $\frac{1}{\sqrt{d}}$, $\frac{1}{\sqrt{c}}$ and $\frac{1}{dc}$ instead, with various combinations of d and c; but the autoencoder failed to learn meaningful codings in other settings. When the scaling factor is too small (e.g. $\frac{1}{dc}$), all relations get almost the same coding; conversely if the factor is too large (e.g. 1), all codings get very close to 0.

The next important rule is to keep a balance between the updates coming from $\nabla\mathcal{L}_1$ and $\nabla\mathcal{L}_2$. We use Stochastic Gradient Descent (SGD) for optimization, and the common practice (Bottou, 2012) is to set the learning rate as

$$\alpha(\tau) := \frac{\eta}{1+\eta\lambda\tau}. \tag{4}$$

Here, η, λ are hyper-parameters and τ is a counter of processed data points. In this work, in order to control the updates in detail to keep a balance, we modify (4) to use a a step counter τ_r for each relation r, counting “number of updates” instead of data points[2]. That is, whenever $\boldsymbol{M}_r$ gets a nonzero update from a gradient calculation, τ_r increases by 1. Furthermore, we use different hyper-parameters for different “types of updates”, namely η_1, λ_1 for updates coming from $\nabla\mathcal{L}_1$, and η_2, λ_2 for updates coming from $\nabla\mathcal{L}_2$. Thus, let Δ_1 be the partial gradient of $\nabla\mathcal{L}_1$, and Δ_2 the partial gradient of $\nabla\mathcal{L}_2$, we update $\boldsymbol{M}_r$ by $\alpha_1(\tau_r)\Delta_1 + \alpha_2(\tau_r)\Delta_2$ at each step, where

$$\alpha_1(\tau_r) := \frac{\eta_1}{1+\eta_1\lambda_1\tau_r}, \quad \alpha_2(\tau_r) := \frac{\eta_2}{1+\eta_2\lambda_2\tau_r}.$$

The rule for setting η_1, λ_1 and η_2, λ_2 is that, η_2 should be much smaller than η_1, because η_1, η_2 control the magnitude of learning rates at the early stage of training, with the autoencoder still largely random and Δ_2 not making much sense; on the other hand, one has to choose λ_1 and λ_2 such that $\|\Delta_1\|/\lambda_1$ and $\|\Delta_2\|/\lambda_2$ are at the same scale, because the learning rates approach $1/(\lambda_1\tau_r)$ and $1/(\lambda_2\tau_r)$ respectively, as the training proceeds. In this way, the autoencoder will not impose random patterns to relation matrices according to its initialization at the early stage, and a balance is kept between $\alpha_1(\tau_r)\Delta_1$ and $\alpha_2(\tau_r)\Delta_2$ later.

But how to estimate $\|\Delta_1\|$ and $\|\Delta_2\|$? It seems that we can approximately calculate their scales from initialization. In this work, we use i.i.d. Gaussians of variance $1/d$ to initialize parameters, so the initial Euclidean norms are $\|\boldsymbol{u}_h\| \approx 1$, $\|\boldsymbol{v}_t\| \approx 1$, $\|\boldsymbol{M}_r\| \approx \sqrt{d}$, and $\|\boldsymbol{B}\boldsymbol{A}\boldsymbol{m}_r\| \approx \sqrt{dc}$. Thus, by calculating $\nabla\mathcal{L}_1$ and $\nabla\mathcal{L}_2$ using (1) and (3), we have approximately

$$\|\Delta_1\| \approx \|\boldsymbol{u}_h\boldsymbol{v}_t^\top\| \approx 1, \quad \text{and}$$
$$\|\Delta_2\| \approx \|\frac{1}{\sqrt{dc}}\boldsymbol{B}\boldsymbol{c}_r\| \approx \frac{1}{\sqrt{dc}}\|\boldsymbol{B}\boldsymbol{A}\boldsymbol{m}_r\| \approx 1.$$

It suggests that, because of the scaling factor $\frac{1}{\sqrt{dc}}$ in (3), we have $\|\Delta_1\|$ and $\|\Delta_2\|$ at the same scale, so we can set $\lambda_1 = \lambda_2$. This might not be a mere coincidence.

4.1 Training the Base Model

Besides the tricks for joint training, we also found settings that significantly improve the base model on KBC, as briefly discussed below. In Sec.6.3, we will show performance gains by these settings using the FB15k-237 validation set.

Normalization It is better to normalize relation matrices to $\|\boldsymbol{M}_r\| = \sqrt{d}$ during training. This might reduce fluctuations in entity vector updates.

Regularizer It is better to minimize $\|\boldsymbol{M}_r^\top\boldsymbol{M}_r - \frac{1}{d}\operatorname{tr}(\boldsymbol{M}_r^\top\boldsymbol{M}_r)I\|$ during training. This regularizer drives $\boldsymbol{M}_r$ toward an orthogonal matrix (Tian et al., 2016) and might reduce fluctuations in entity vector updates. As a result, all relation matrices trained in this work are very close to orthogonal.

Initialization Instead of pure Gaussian, it is better to initialize matrices as $(I+G)/2$, where G is random. The identity matrix I helps passing information from head to tail (Tian et al., 2016).

Negative Sampling Instead of a unigram distribution, it is better to use a *uniform* distribution for generating noises. This is somehow counter-intuitive compared to training word embeddings.

5 Related Works

KBs have a wide range of applications (Berant et al., 2013; Hixon et al., 2015; Nickel et al., 2016a) and KBC has inspired a huge amount of research (Bordes et al., 2013; Riedel et al., 2013; Socher et al., 2013; Wang et al., 2014b,a; Xiao et al., 2016; Nguyen et al., 2016; Toutanova et al., 2016; Das et al., 2017; Hayashi and Shimbo, 2017).

[2]Similarly, we set separate step counters for all head and tail entities, and the autoencoder as well.

Among the previous works, TransE (Bordes et al., 2013) is the classic method which represents a relation as a translation of the entity vector space, and is partially inspired by Mikolov et al. (2013)'s vector arithmetic method of solving word analogy tasks. Although competitive in KBC, it is speculated that this method is well-suited for 1-to-1 relations but might be too simple to represent N-to-N relations accurately(Wang et al., 2017). Thus, extensions such as TransR (Lin et al., 2015b) and STransE (Nguyen et al., 2016) are proposed to map entities into a relation-specific vector space before translation. The ITransF model (Xie et al., 2017) further enhances this approach by imposing a hard constraint that the relation-specific maps should be linear combinations of a small number of prototypical matrices. Our work inherits the same motivation with ITransF in terms of promoting parameter-sharing among relations.

On the other hand, the base model used in this work originates from RESCAL (Nickel et al., 2011), in which relations are naturally represented as analogue to the adjacency matrices (Sec.2). Further developments include HolE (Nickel et al., 2016b) and ConvE (Dettmers et al., 2018) which improve this approach in terms of parameter-efficiency, by introducing low dimension factorizations of the matrices. We inherit the basic model of RESCAL but draw additional training techniques from Tian et al. (2016), and show that the base model already can achieve near state-of-the-art performance (Sec.6.1,6.3). This sends a message similar to Kadlec et al. (2017), saying that training tricks might be as important as model designs.

Nevertheless, we emphasize the novelty of this work in that the previous models mostly achieve dimension reduction by imposing some pre-designed hard constraints (Bordes et al., 2013; Yang et al., 2015; Trouillon et al., 2016; Nickel et al., 2016b; Xie et al., 2017; Dettmers et al., 2018), whereas the constraints themselves are not learned from data; in contrast, our approach by jointly training an autoencoder does not impose any explicit hard constraints, so it leads to more flexible modeling.

Moreover, we additionally focus on leveraging composition in KBC. Although this idea has been frequently explored before (Guu et al., 2015; Neelakantan et al., 2015; Lin et al., 2015a), our discussion about the concept of compositional constraints and its connection to dimension reduction has not been addressed similarly in previous research. In experiments, we will show (Sec.6.2,6.3) that joint training with an autoencoder indeed helps finding compositional constraints and benefits from compositional training.

Autoencoders have been used solo for learning distributed representations of syntactic trees (Socher et al., 2011), words and images (Silberer and Lapata, 2014), or semantic roles (Titov and Khoddam, 2015). It is also used for pretraining other deep neural networks (Erhan et al., 2010). However, when combined with other models, the learning of autoencoders, or more generally *sparse codings* (Rubinstein et al., 2010), is usually conveyed in an alternating manner, fixing one part of the model while optimizing the other, such as in Xie et al. (2017). To our knowledge, joint training with an autoencoder is not widely used previously for reducing dimensionality.

Jointly training an autoencoder is not simple because it takes non-stationary inputs. In this work, we modified SGD so that it shares traits with some modern optimization algorithms such as Adagrad (Duchi et al., 2011), in that they both set different learning rates for different parameters. While Adagrad sets them adaptively by keeping track of gradients for all parameters, our modification of SGD is more efficient and allows us to grasp a rough intuition about which parameter gets how much update. We believe our techniques and findings in joint training with an autoencoder could be helpful to reducing dimensionality and improving interpretability in other neural network architectures as well.

6 Experiments

We evaluate on standard KBC datasets, including WN18 and FB15k (Bordes et al., 2013), WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova and Chen, 2015). The statistical information of these datasets are shown in Table 1.

WN18 collects word relations from WordNet (Miller, 1995), and FB15k is taken from Freebase (Bollacker et al., 2008); both have filtered out low frequency entities. However, it is reported in Toutanova and Chen (2015) that both WN18 and FB15k have information leaks because the inverses of some test triples appear in the training set. FB15k-237 and WN18RR fix this problem by deleting such triples from training and test data. In this work, we do evaluate on WN18 and FB15k, but our models are mainly tuned on FB15k-237.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	#Train	#Valid	#Test
WN18	40,943	18	141,442	5,000	5,000
FB15k	14,951	1,345	483,142	50,000	59,071
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466

Table 1: Statistical information of the KBC datasets. $|\mathcal{E}|$ and $|\mathcal{R}|$ denote the number of entities and relation types, respectively; #Train, #Valid, and #Test are the numbers of triples in the training, validation, and test sets, respectively.

For all datasets, we set the dimension $d = 256$ and $c = 16$, the SGD hyper-parameters $\eta_1 = 1/64$, $\eta_2 = 2^{-14}$ and $\lambda_1 = \lambda_2 = 2^{-14}$. The training batch size is 32 and the triples in each batch share the same head entity. We compare the base model (BASE) to our joint training with an autoencoder model (JOINT), and the base model with compositional training (BASE+COMP) to our joint model with compositional training (JOINT+COMP). When compositional training is enabled (BASE+COMP, JOINT+COMP), we use random walk to sample paths of length $1 + X$, where X is drawn from a Poisson distribution of mean $\lambda = 1.0$.

For any incomplete triple $\langle h, r, ?\rangle$ in KBC test, we calculate a score $s(h, r, e)$ from (1), for every entity $e \in \mathcal{E}$ such that $\langle h, r, e\rangle$ *does not appear in any of the training, validation, or test sets* (Bordes et al., 2013). Then, the calculated scores together with $s(h, r, t)$ for the gold triple is converted to ranks, and the rank of the gold entity t is used for evaluation. Evaluation metrics include Mean Rank (MR), Mean Reciprocal Rank (MRR), and Hits at 10 (H10). Lower MR, higher MRR, and higher H10 indicate better performance.

We consult MR and MRR on validation sets to determine training epochs; we stop training when both MR and MRR have stopped improving.

6.1 KBC Results

The results are shown in Table 2. We found that joint training with an autoencoder mostly improves performance, and the improvement becomes more clear when compositional training is enabled (i.e., JOINT $\geq$ BASE and JOINT+COMP $>$ BASE+COMP). This is convincing because generally, joint training contributes with its regularizing effects, and drastic improvements are less expected[3]. When compositional training is enabled, the system usually achieves better MR, though not always improves in other measures. The performance gains are more obvious on the WN18RR and FB15k-237 datasets, possibly because WN18 and FB15k contain a lot of easy instances that can be solved by a simple rule (Dettmers et al., 2018).

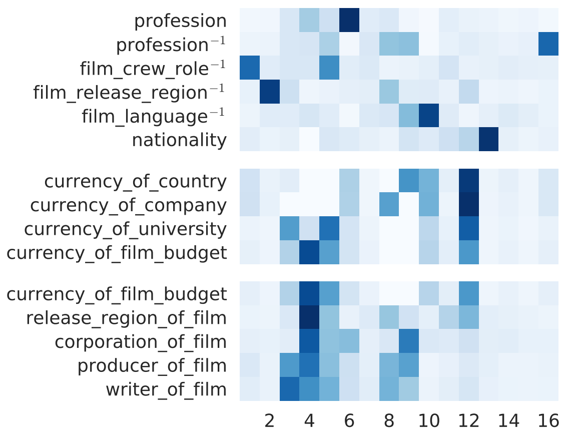

Figure 2: Examples of relation codings learned from FB15k-237. Each row shows a 16 dimension vector encoding a relation. Vectors are normalized such that their entries sum to 1.

Furthermore, the numbers demonstrated by our joint and base models are among the strongest in the literature. We have conducted re-experiments of several representative algorithms, and also compare with state-of-the-art published results. For re-experiments, we use Lin et al. (2015b)'s implementation[4] of TransE (Bordes et al., 2013) and TransR, which represent relations as vector translations; and Nickel et al. (2016b)'s implementation[5] of RESCAL (Nickel et al., 2011) and HolE, where RESCAL is most similar to the BASE model and HolE is a more parameter-efficient variant. We experimented with the default settings, and found that our models outperform most of them.

Among the published results, STransE (Nguyen et al., 2016) and ITransF (Xie et al., 2017) are more complicated versions of TransR, achieving the previous highest MR on WN18 but are outperformed by our JOINT+COMP model. ITransF is most similar to our JOINT model in that they both learn sparse codings for relations. On WN18RR and FB15k-237, Dettmers et al. (2018)'s report of ComplEx

[3]The source code and trained models are publicly released at `https://github.com/tianran/glimvec`, where we also show the mean performance and deviations of multiple random initializations, to give a more complete picture.

[4]`https://github.com/thunlp/KB2E`

[5]`https://github.com/mnick/holographic-embeddings`

Model	WN18		FB15k		WN18RR			FB15k-237		
	MR	H10	MR	H10	MR	MRR	H10	MR	MRR	H10
JOINT	**277**	**95.8**	**53**	**82.5**	**4233**	**.461***	**53.4**	**212**	.336	**52.3***
BASE	286	**95.8**	**53**	**82.5**	4371	.459	52.9	215	**.337***	**52.3***
JOINT+COMP	**191***	**94.8**	**53**	**69.7**	**2268***	**.343**	**54.8***	**197***	**.331**	**51.6**
BASE+COMP	195	**94.8**	54	69.4	2447	.310	54.1	203	.328	51.5
TransE (Bordes et al., 2013)	292	92.0	**66**	70.4	4311	.202	45.6	**278**	.236	41.6
TransR (Lin et al., 2015b)	**281**	93.6	76	**74.4**	**4222**	.210	**47.1**	320	**.282**	**45.9**
RESCAL (Nickel et al., 2011)	911	58.0	163	41.0	9689	.105	20.3	457	.178	31.9
HolE (Nickel et al., 2016b)	724	**94.3**	293	66.8	8096	**.376**	40.0	1172	.169	30.9
STransE (Nguyen et al., 2016)	206	93.4	69	79.9	-	-	-	-	-	-
ITransF (Xie et al., 2017)	**205**	94.2	65	81.0	-	-	-	-	-	-
ComplEx (Trouillon et al., 2016)	-	94.7	-	84.0	**5261**	.44	**51**	339	.247	42.8
Ensemble DistMult (Kadlec et al., 2017)	457	95.0	35.9	90.4	-	-	-	-	-	-
IRN (Shen et al., 2017)	249	95.3	38	**92.7***	-	-	-	-	-	-
ConvE (Dettmers et al., 2018)	504	95.5	64	87.3	5277	**.46**	48	**246**	**.316**	**49.1**
R-GCN+ (Schlichtkrull et al., 2017)	-	**96.4***	-	84.2	-	-	-	-	.249	41.7
ProjE (Shi and Weninger, 2017)	-	-	**34***	88.4	-	-	-	-	-	-

Table 2: KBC results on the WN18, FB15k, WN18RR, and FB15k-237 datasets. The first and second sectors compare our joint to the base models with and without compositional training, respectively; the third sector shows our re-experiments and the fourth shows previous published results. Bold numbers are the best in each sector, and $(^*)$ indicates the best of all.

(Trouillon et al., 2016) and ConvE were previously the best results. Our models mostly outperform them. Other results include Kadlec et al. (2017)'s simple but strong baseline and several recent models (Schlichtkrull et al., 2017; Shi and Weninger, 2017; Shen et al., 2017) which achieve best results on FB15k or WN18 in some measure. Our models have comparable results.

6.2 Intuition and Insight

What does the autoencoder look like? How does joint training affect relation matrices? We address these questions by analyses showing that **(i)** the autoencoder learns sparse and interpretable codings of relations, **(ii)** the joint training drives relation matrices toward a low dimension manifold, and **(iii)** it helps discovering compositional constraints.

Sparse Coding and Interpretability

Due to the ReLU function in (2), our autoencoder learns sparse coding, with most relations having large code values at only two or three dimensions. This sparsity makes it easy to find patterns in the model that to some extent explain the semantics of relations. Figure 2 shows some examples.

In the first group of Figure 2, we show a small number of relations that are almost always assigned a near one-hot coding, regardless of initialization. These are high frequency relations joining two large categories (e.g. film and language), which probably constitute the skeleton of a KB.

In the second group, we found the 12th dimension strongly correlates with `currency`; and in the third group, we found the 4th dimension strongly correlates with `film`. As for the relation `currency_of_film_budget`, it has large code values at both dimensions. This kind of relation clustering also seems independent of initialization. Intuitively, it shows that the autoencoder may discover similarities between relations and promote indirect parameter sharing among them. Yet, as the autoencoder only reconstructs *approximations* of relation matrices but never constrain them to be exactly equal to the original, relation matrices with very similar codings may still differ considerably. For example, `producer_of_film` and `writer_of_film` have codings of cosine similarity 0.973, but their relation matrices only have[6] a cosine similarity 0.338.

Low dimension manifold

In order to visualize the relation matrices learned by our joint and base models, we use UMAP[7] (McInnes and Healy, 2018) to embed M_r into a 2D plane[8]. We use relation matrices trained on

[6]Cosine similarity 0.338 is still high for matrices, due to the high dimensionality of their parameter space.

[7]`https://github.com/lmcinnes/umap`

[8]UMAP is a recently proposed manifold learning algorithm based on the fuzzy topological structure. We also tried

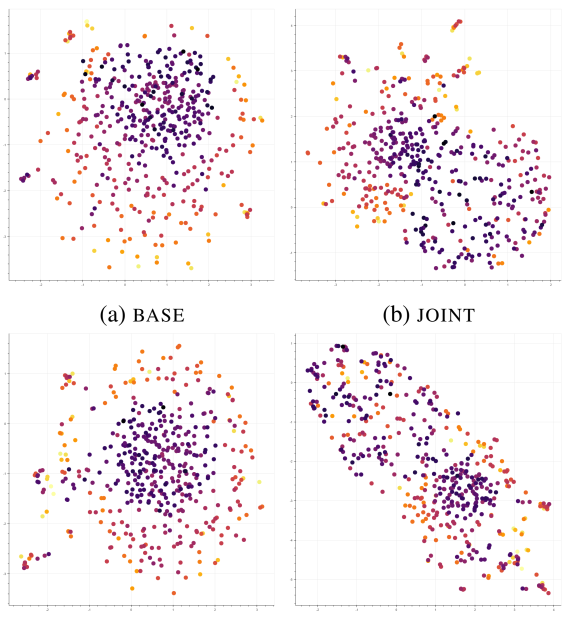
(a) BASE (b) JOINT (c) BASE+COMP (d) JOINT+COMP

Figure 3: By UMAP, relation matrices are embedded into a 2D plane. Colors show frequencies of relations; and lighter color means more frequent.

Model	MR	MRR
JOINT+COMP	**130±27**	**.0481±.0090**
BASE+COMP	150±3	.0280±.0010
RANDOMM2	181±19	.0356±.0100

Table 3: Performance at discovering compositional constraints extracted from FB15k-237

FB15k-237, and compare models trained by the same number of epochs. The results are shown in Figure 3.

We can see that Figure 3a and Figure 3c are mostly similar, with high frequency relations scattered randomly around a low frequency cluster, suggesting that they come from various directions of a high dimension space, with frequent relations probably being pulled further by the training updates. On the other hand, in Figure 3b and Figure 3d we found less frequent relations being clustered with frequent ones, and multiple traces of low dimension structures. It suggests that joint training with an autoencoder indeed drives relations toward a low dimension manifold. In addition, Figure 3d shows different structures against Figure 3b, which we conjecture could be related to compositional constraints discovered by compositional training.

Compositional constraints

In order to directly evaluate a model's ability to find compositional constraints, we extracted from FB15k-237 a list of $(r_1/r_2, r_3)$ pairs such that r_1/r_2 matches r_3. Formally, the list is constructed as below. For any relation r, we define a *content set* $C(r)$ as the set of (h,t) pairs such that $\langle h,r,t\rangle$ is a fact in the KB. Similarly, we define $C(r_1/r_2)$ as the set of (h,t) pairs such that $\langle h, r_1/r_2, t\rangle$ is a path. We regard $(r_1/r_2, r_3)$ as a compositional constraint if their content sets are similar; that is, if $|C(r_1/r_2) \cap C(r_3)| \geq 50$ and the Jaccard similarity between $C(r_1/r_2)$ and $C(r_3)$ is ≥ 0.4. Then, after filtering out degenerated cases such as $r_1 = r_3$ or $r_2 = r_1^{-1}$, we obtained a list of 154 compositional constraints, e.g. (`currency_of_country/country_of_film`, `currency_of_film_budget`).

For each compositional constraint $(r_1/r_2, r_3)$ in the list, we take the matrices $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and $\boldsymbol{M}_3$ corresponding to r_1, r_2 and r_3 respectively, and rank $\boldsymbol{M}_3$ according to its cosine similarity with $\boldsymbol{M}_1\boldsymbol{M}_2$, among all relation matrices. Then, we calculate MR and MRR for evaluation. We compare the JOINT+COMP model to BASE+COMP, as well as a randomized baseline where M_2 is selected randomly from the relation matrices in JOINT+COMP instead (RANDOMM2). The results are shown in Table 3. We have evaluated 5 different random initializations for each model, trained by the same number of epochs, and we report the mean and standard deviation. We verify that JOINT+COMP performs better than BASE+COMP, indicating that joint training with an autoencoder indeed helps discovering compositional constraints. Furthermore, the random baseline RANDOMM2 tests a hypothesis that joint training might be just clustering M_3 and M_1 here, to the extent that M_3 and M_1 are so close that even a random M_2 can give the correct answer; but as it turns out, JOINT+COMP largely outperforms RANDOMM2, excluding this possibility. Thus, joint training performs better not simply because it clusters relation matrices; it learns compositions indeed.

6.3 Losses and Gains

In the KBC task, where are the losses and what are the gains of different settings? With additional evaluations, we show **(i)** some crucial settings for the base model, and **(ii)** joint training with an autoencoder benefits more from compositional training.

t-SNE (van der Maaten and Hinton, 2008) but found UMAP more insightful.

Settings	MR	MRR	H10
BASE	**214**	**.338**	**52.5**
no normalization	309	.326	49.9
no regularizer	400	.328	51.3
pure Gaussian	221	.336	52.1
unigram distribution	215	.324	50.6

Table 4: Ablation of the four settings of the base model as described in Sec.4.1

Crucial settings for the base model

It is noteworthy that our base model already achieves strong results. This is due to several detailed but crucial settings as we discussed in Sec.4.1; Table 4 shows their gains on the FB15k-237 validation data. The most dramatic improvement comes from the regularizer that drives matrices to orthogonal.

Gains with compositional training

One can force a model to focus more on (longer) compositions of relations, by sampling longer paths in compositional training. Since joint training with an autoencoder helps discovering compositional constraints, we expect it to be more helpful when the sampled paths are longer. In this work, path lengths are sampled from a Poisson distribution, we thus vary the mean λ of the Poisson to control the strength of compositional training. The results on FB15k-237 are shown in Table 5.

We can see that, as λ gets larger, MR improves much but MRR slightly drops. It suggests that in FB15k-237, composition of relations might mainly help finding more appropriate candidates for a missing entity, rather than pinpointing a correct one. Yet, joint training improves base models even more as the paths get longer, especially in MR. It further supports our conjecture that joint training with an autoencoder may strongly interact with compositional training.

Model	λ	Valid			Test		
		MR	MRR	H10	MR	MRR	H10
BASE	0	209	.341	52.9	215	.337	52.3
JOINT	0	+1	-.001	-.2	**-3**	-.001	0
BASE	0.5	204	.337	52.2	211	.332	51.7
JOINT	0.5	**-3**	**+.002**	**+.1**	+1	**+.002**	**+.2**
BASE	1.0	191	.334	52.0	203	.328	51.5
JOINT	1.0	**-5**	**+.002**	-.1	**-6**	**+.003**	**+.1**

Table 5: Evaluation of BASE and gains by JOINT, on FB15k-237 with different strengths of compositional training. Bold numbers are improvements.

7 Conclusion

We have investigated a dimension reduction technique which trains a KB embedding model jointly with an autoencoder. We have developed new training techniques and achieved state-of-the-art results on several KBC tasks with strong improvements in Mean Rank. Furthermore, we have shown that the autoencoder learns low dimension sparse codings that can be easily explained; the joint training technique drives high-dimensional data toward low dimension manifolds; and the reduction of dimensionality may interact strongly with composition, help discovering compositional constraints and benefit from compositional training. We believe these findings provide insightful understandings of KB embedding models and might be applied to other neural networks beyond the KBC task.

Acknowledgments

This work was supported by JST CREST Grant Number JPMJCR1301, Japan. We thank Pontus Stenetorp, Makoto Miwa, and the anonymous reviewers for many helpful advices and comments.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, pages 722–735.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information*

Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., pages 2787–2795.

Léon Bottou. 2012. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 132–141, Valencia, Spain. Association for Computational Linguistics.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Dumitru Erhan, Yoshua Bengio, Aaron C. Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660.

Michael Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327, Lisbon, Portugal. Association for Computational Linguistics.

Katsuhiko Hayashi and Masashi Shimbo. 2017. On the equivalence of holographic and complex embeddings for link prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 554–559, Vancouver, Canada. Association for Computational Linguistics.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 851–861, Denver, Colorado. Association for Computational Linguistics.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74, Vancouver, Canada. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714, Lisbon, Portugal. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2181–2187.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

L. McInnes and J. Healy. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 156–166, Beijing, China. Association for Computational Linguistics.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–466, San Diego, California. Association for Computational Linguistics.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016b. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 809–816, USA. Omnipress.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia. Association for Computational Linguistics.

R. Rubinstein, A. M. Bruckstein, and M. Elad. 2010. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *CoRR*, abs/1703.06103.

Yelong Shen, Po-Sen Huang, Ming-Wei Chang, and Jianfeng Gao. 2017. Modeling large-scale structured relationships with shared memory for knowledge base completion. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 57–68, Vancouver, Canada. Association for Computational Linguistics.

Baoxu Shi and Tim Weninger. 2017. Proje: Embedding projection for knowledge graph completion. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 1236–1242.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–732, Baltimore, Maryland. Association for Computational Linguistics.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 926–934.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Learning semantically and additively compositional distributional representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1277–1287, Berlin, Germany. Association for Computational Linguistics.

Ivan Titov and Ehsan Khoddam. 2015. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–10, Denver, Colorado. Association for Computational Linguistics.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1434–1444, Berlin, Germany. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2071–2080.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014a. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1591–1601, Doha, Qatar. Association for Computational Linguistics.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1112–1119.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1315–1321.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. 2017. An interpretable knowledge transfer model for knowledge base completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 950–962, Vancouver, Canada. Association for Computational Linguistics.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the 3rd International Conference on Learning Representations*, pages 1–12.

A Out-of-vocabulary Entities in KBC

Occasionally, a KBC test set may contain entities that never appear in the training data. Such out-of-vocabulary (OOV) entities pose a challenge to KBC systems; while some systems address this issue by explicitly learn an OOV entity vector (Dettmers et al., 2018), our approach is described below. For an incomplete triple $\langle h, r, ?\rangle$ in the test, if h is OOV, we replace it with the most frequent entity that has ever appeared as a head of relation r in the training data. If the gold tail entity is OOV, we use the zero vector for computing the score and the rank of the gold entity.

Usually, OOV entities are rare and negligible in evaluation; except for the WN18RR test data which contains about 6.7% triples with OOV entities. Here, we also report adjusted scores on WN18RR in the setting that all triples with OOV entities are removed from the test. The results are shown in Table 6.

Model	MR	MRR	H10
JOINT	**3317**	**.493**	**57.2**
BASE	3435	.492	56.7
JOINT+COMP	**1507**	**.367**	**58.7**
BASE+COMP	1629	.332	58.0

Table 6: Adjusted scores on WN18RR.