Traversing Knowledge Graphs in Vector Space

Kelvin Guu
Stanford University
kguu@stanford.edu

John Miller
Stanford University
millerjp@stanford.edu

Percy Liang
Stanford University
pliang@cs.stanford.edu

Abstract

Path queries on a knowledge graph can be used to answer compositional questions such as "*What languages are spoken by people living in Lisbon?*". However, knowledge graphs often have missing facts (edges) which disrupts path queries. Recent models for knowledge base completion impute missing facts by embedding knowledge graphs in vector spaces. We show that these models can be recursively applied to answer path queries, but that they suffer from cascading errors. This motivates a new "compositional" training objective, which dramatically improves all models' ability to answer path queries, in some cases more than doubling accuracy. On a standard knowledge base completion task, we also demonstrate that compositional training acts as a novel form of structural regularization, reliably improving performance across all base models (reducing errors by up to 43%) and achieving new state-of-the-art results.

1 Introduction

Broad-coverage knowledge bases such as Freebase (Bollacker et al., 2008) support a rich array of reasoning and question answering applications, but they are known to suffer from incomplete coverage (Min et al., 2013). For example, as of May 2015, Freebase has an entity Tad Lincoln (Abraham Lincoln's son), but does not have his ethnicity. An elegant solution to incompleteness is using vector space representations: Controlling the dimensionality of the vector space forces generalization to new facts (Nickel et al., 2011; Nickel et al., 2012; Socher et al., 2013; Riedel et al., 2013; Neelakantan et al., 2015). In the example, we would hope to infer Tad's ethnicity from the ethnicity of his parents.

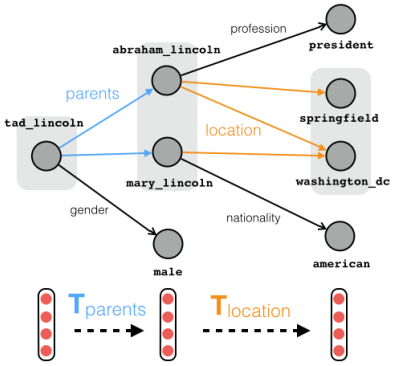

Figure 1: We propose performing path queries such as `tad_lincoln/parents/location` ("*Where are Tad Lincoln's parents located?*") in a parallel low-dimensional vector space. Here, entity sets (boxed) are represented as real vectors, and edge traversal is driven by vector-to-vector transformations (e.g., matrix multiplication).

However, what is missing from these vector space models is the original strength of knowledge bases: the ability to support compositional queries (Ullman, 1985). For example, we might ask what the ethnicity of Abraham Lincoln's daughter would be. This can be formulated as a path query on the knowledge graph, and we would like a method that can answer this efficiently, while generalizing over missing facts and even missing or hypothetical entities (Abraham Lincoln did not in fact have a daughter).

In this paper, we present a scheme to answer path queries on knowledge bases by "compositionalizing" a broad class of vector space models that have been used for knowledge base completion (see Figure 1). At a high level, we interpret the base vector space model as implementing a soft edge traversal operator. This operator can then be recursively applied to predict paths. Our interpretation suggests a new *compositional training* objective that encourages better modeling of

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 318–327,
Lisbon, Portugal, 17-21 September 2015.

paths. Our technique is applicable to a broad class of *composable* models that includes the bilinear model (Nickel et al., 2011) and TransE (Bordes et al., 2013).

We have two key empirical findings: First, we show that compositional training enables us to answer path queries up to at least length 5 by substantially reducing cascading errors present in the base vector space model. Second, we find that somewhat surprisingly, compositional training also improves upon state-of-the-art performance for knowledge base completion, which is a special case of answering unit length path queries. Therefore, compositional training can also be seen as a new form of structural regularization for existing models.

2 Task

We now give a formal definition of the task of answering path queries on a knowledge base. Let $\mathcal{E}$ be a set of entities and $\mathcal{R}$ be a set of binary relations. A knowledge graph $\mathcal{G}$ is defined as a set of triples of the form (s, r, t) where $s, t \in \mathcal{E}$ and $r \in \mathcal{R}$. An example of a triple in Freebase is $(\texttt{tad\_lincoln}, \texttt{parents}, \texttt{abraham\_lincoln})$.

A path query q consists of an initial *anchor entity,* s, followed by a sequence of relations to be traversed, $p = (r_1, \ldots, r_k)$. The answer or denotation of the query, $[\![q]\!]$, is the set of all entities that can be reached from s by traversing p. Formally, this can be defined recursively:

$$[\![s]\!] \stackrel{\text{def}}{=} \{s\}, \tag{1}$$

$$[\![q/r]\!] \stackrel{\text{def}}{=} \{t : \exists s \in [\![q]\!], (s, r, t) \in \mathcal{G}\}. \tag{2}$$

For example, `tad_lincoln/parents/location` is a query q that asks: "*Where did Tad Lincoln's parents live?*".

For evaluation (see Section 5 for details), we define the set of candidate answers to a query $\mathcal{C}(q)$ as the set of all entities that "type match", namely those that participate in the final relation of q at least once; and let $\mathcal{N}(q)$ be the incorrect answers:

$$\mathcal{C}\left(s/r_1/\cdots/r_k\right) \stackrel{\text{def}}{=} \{t \mid \exists e, (e, r_k, t) \in \mathcal{G}\} \tag{3}$$

$$\mathcal{N}\left(q\right) \stackrel{\text{def}}{=} \mathcal{C}\left(q\right) \setminus [\![q]\!]. \tag{4}$$

Knowledge base completion. Knowledge base completion (KBC) is the task of predicting whether a given edge (s, r, t) belongs in the graph or not. This can be formulated as a path query $q = s/r$ with candidate answer t.

3 Compositionalization

In this section, we show how to compositionalize existing KBC models to answer path queries. We start with a motivating example in Section 3.1, then present the general technique in Section 3.2. This suggests a new compositional training objective, described in Section 3.3. Finally, we illustrate the technique for several more models in Section 3.4, which we use in our experiments.

3.1 Example

A common vector space model for knowledge base completion is the *bilinear model* (Nickel et al., 2011). In this model, we learn a vector $x_e \in \mathbb{R}^d$ for each entity $e \in \mathcal{E}$ and a matrix $W_r \in \mathbb{R}^{d \times d}$ for each relation $r \in \mathcal{R}$. Given a query s/r (asking for the set of entities connected to s via relation r), the bilinear model scores how likely $t \in [\![s/r]\!]$ holds using

$$\text{score}(s/r, t) = x_s^\top W_r x_t. \tag{5}$$

To motivate our compositionalization technique, take $d = |\mathcal{E}|$ and suppose W_r is the adjacency matrix for relation r and entity vector x_e is the indicator vector with a 1 in the entry corresponding to entity e. Then, to answer a path query $q = s/r_1/\ldots/r_k$, we would then compute

$$\text{score}(q, t) = x_s^\top W_{r_1} \ldots W_{r_k} x_t. \tag{6}$$

It is easy to verify that the score counts the number of unique paths between s and t following relations $r_1/\ldots/r_k$. Hence, any t with positive score is a correct answer ($[\![q]\!] = \{t : \text{score}(q, t) > 0\}$).

Let us interpret (6) recursively. The model begins with an entity vector x_s, and sequentially applies *traversal operators* $\mathbb{T}_{r_i}(v) = v^\top W_{r_i}$ for each r_i. Each traversal operation results in a new "set vector" representing the entities reached at that point in traversal (corresponding to the nonzero entries of the set vector). Finally, it applies the *membership operator* $\mathbb{M}(v, x_t) = v^\top x_t$ to check if $t \in [\![s/r_1/\ldots/r_k]\!]$. Writing graph traversal in this way immediately suggests a useful generalization: take d much smaller than $|\mathcal{E}|$ and learn the parameters W_r and x_e.

3.2 General technique

The strategy used to extend the bilinear model of (5) to the compositional model in (6) can be applied to any *composable* model: namely, one that

has a scoring function of the form:

$$\text{score}(s/r, t) = \mathbb{M}(\mathbb{T}_r(x_s), x_t) \tag{7}$$

for some choice of membership operator $\mathbb{M} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and traversal operator $\mathbb{T}_r : \mathbb{R}^d \to \mathbb{R}^d$.

We can now define the *vector denotation* of a query $[\![q]\!]_{\text{V}}$ analogous to the definition of $[\![q]\!]$ in (1) and (2):

$$[\![s]\!]_{\text{V}} \stackrel{\text{def}}{=} x_s, \tag{8}$$

$$[\![q/r]\!]_{\text{V}} \stackrel{\text{def}}{=} \mathbb{T}_r\left([\![q]\!]_{\text{V}}\right). \tag{9}$$

The score function for a compositionalized model is then

$$\text{score}(q, t) = \mathbb{M}([\![q]\!]_{\text{V}}, [\![t]\!]_{\text{V}}). \tag{10}$$

We would like $[\![q]\!]_{\text{V}}$ to approximately represent the set $[\![q]\!]$ in the sense that for every $e \in [\![q]\!]$, $\mathbb{M}\left([\![q]\!]_{\text{V}}, [\![e]\!]_{\text{V}}\right)$ is larger than the values for $e \notin [\![q]\!]$. Of course it is not possible to represent all sets perfectly, but in the next section, we present a training objective that explicitly optimizes $\mathbb{T}$ and $\mathbb{M}$ to preserve path information.

3.3 Compositional training

The score function in (10) naturally suggests a new compositional training objective. Let $\{(q_i, t_i)\}_{i=1}^N$ denote a set of path query training examples with path lengths ranging from 1 to L. We minimize the following max-margin objective:

$$J(\Theta) = \sum_{i=1}^{N} \sum_{t' \in \mathcal{N}(q_i)} \left[1 - \text{margin}(q_i, t_i, t')\right]_+,$$

$$\text{margin}(q, t, t') = \text{score}(q, t) - \text{score}(q, t'),$$

where the parameters are the membership operator, the traversal operators, and the entity vectors:

$$\Theta = \{\mathbb{M}\} \cup \{\mathbb{T}_r : r \in \mathcal{R}\} \cup \left\{x_e \in \mathbb{R}^d : e \in \mathcal{E}\right\}.$$

This objective encourages the construction of "set vectors": because there are path queries of different lengths and types, the model must learn to produce an accurate set vector $[\![q]\!]_{\text{V}}$ after any sequence of traversals. Another perspective is that each traversal operator is trained such that its transformation preserves information in the set vector which might be needed in subsequent traversal steps.

In contrast, previously proposed training objectives for knowledge base completion only train on queries of path length 1. We will refer to this special case as *single-edge training*.

In Section 5, we show that compositional training leads to substantially better results for both path query answering and knowledge base completion. In Section 6, we provide insight into why.

3.4 Other composable models

There are many possible candidates for $\mathbb{T}$ and $\mathbb{M}$. For example, $\mathbb{T}$ could be one's favorite neural network mapping from $\mathbb{R}^d$ to $\mathbb{R}^d$. Here, we focus on two composable models that were both recently shown to achieve state-of-the-art performance on knowledge base completion.

TransE. The TransE model of Bordes et al. (2013) uses the scoring function

$$\text{score}(s/r, t) = -\|x_s + w_r - x_t\|_2^2. \tag{11}$$

where x_s, w_r and x_t are all d-dimensional vectors.

In this case, the model can be expressed using membership operator

$$\mathbb{M}(v, x_t) = -\|v - x_t\|_2^2 \tag{12}$$

and traversal operator $\mathbb{T}_r(x_s) = x_s + w_r$. Hence, TransE can handle a path query $q = s/r_1/r_2/\cdots/r_k$ using

$$\text{score}(q, t) = -\|x_s + w_{r_1} + \cdots + w_{r_k} - x_t\|_2^2.$$

We visualize the compositional TransE model in Figure 2.

Bilinear-Diag. The Bilinear-Diag model of Yang et al. (2015) is a special case of the bilinear model with the relation matrices constrained to be diagonal. Alternatively, the model can be viewed as a variant of TransE with multiplicative interactions between entity and relation vectors.

Not all models can be compositionalized. It is important to point out that some models are not naturally composable—for example, the latent feature model of Riedel et al. (2013) and the neural tensor network of Socher et al. (2013). These approaches have scoring functions which combine s, r and t in a way that does not involve an intermediate vector representing s/r alone without t, so they do not decompose according to (7).

		WordNet	Freebase
Relations		11	13
Entities		38,696	75,043
Base	Train	112,581	316,232
	Test	10,544	23,733
Paths	Train	2,129,539	6,266,058
	Test	46,577	109,557

Table 1: WordNet and Freebase statistics for base and path query datasets.

3.5 Implementation

We use AdaGrad (Duchi et al., 2010) to optimize $J(\Theta)$, which is in general non-convex. Initialization scale, mini-batch size and step size were cross-validated for all models. We initialize all parameters with i.i.d. Gaussians of variance 0.1 in every entry, use a mini-batch size of 300 examples, and a step size in $[0.001, 0.1]$ (chosen via cross-validation) for all of the models. For each example q, we sample 10 negative entities $t' \in \mathcal{N}(q)$. During training, all of the entity vectors are constrained to lie on the unit ball, and we clipped the gradients to the median of the observed gradients if the update exceeded 3 times the median.

We first train on path queries of length 1 until convergence and then train on all path queries until convergence. This guarantees that the model masters basic edges before composing them to form paths. When training on path queries, we explicitly parameterize inverse relations. For the bilinear model, we initialize $W_{r^{-1}}$ with $W_r^\top$. For TransE, we initialize $w_{r^{-1}}$ with $-w_r$. For Bilinear-Diag, we found initializing $w_{r^{-1}}$ with the exact inverse $1/w_r$ is numerically unstable, so we instead randomly initialize $w_{r^{-1}}$ with i.i.d Gaussians of variance 0.1 in every entry. Additionally, for the bilinear model, we replaced the sum over $\mathcal{N}(q_i)$ in the objective with a max since it yielded slightly higher accuracy. Our models are implemented using Theano (Bastien et al., 2012; Bergstra et al., 2010).

4 Datasets

In Section 4.1, we describe two standard knowledge base completion datasets. These consist of single-edge queries, so we call them *base datasets*. In Section 4.2, we generate path query datasets from these base datasets.

4.1 Base datasets

Our experiments are conducted using the subsets of WordNet and Freebase from Socher et al. (2013). The statistics of these datasets and their splits are given in Table 1.

The WordNet and Freebase subsets exhibit substantial differences that can influence model performance. The Freebase subset is almost bipartite with most of the edges taking the form (s, r, t) for some *person* s, relation r and property t. In WordNet, both the source and target entities are arbitrary words.

Both the raw WordNet and Freebase contain many relations that are almost perfectly correlated with an inverse relation. For example, WordNet contains both `has_part` and `part_of`, and Freebase contains both `parents` and `children`. At test time, a query on an edge (s, r, t) is easy to answer if the inverse triple (t, r^{-1}, s) was observed in the training set. Following Socher et al. (2013), we account for this by excluding such "trivial" queries from the test set.

4.2 Path query datasets

Given a base knowledge graph, we generate path queries by performing random walks on the graph. If we view compositional training as a form of regularization, this approach allows us to generate extremely large amounts of auxiliary training data. The procedure is given below.

Let $\mathcal{G}_{\text{train}}$ be the training graph, which consists only of the edges in the training set of the base dataset. We then repeatedly generate training examples with the following procedure:

1. Uniformly sample a path length $L \in \{1, \ldots, L_{\max}\}$, and uniformly sample a starting entity $s \in \mathcal{E}$.

2. Perform a random walk beginning at entity s and continuing L steps.

   (a) At step i of the walk, choose a relation r_i uniformly from the set of relations incident on the current entity e.

   (b) Choose the next entity uniformly from the set of entities reachable via r_i.

3. Output a query-answer pair, (q, t), where $q = s/r_1/\cdots/r_L$ and t is the final entity of the random walk.

In practice, we do not sample paths of length 1 and instead directly add all of the edges from $\mathcal{G}_{\text{train}}$ to the path query dataset.

To generate a path query test set, we repeat the above procedure except using the graph $\mathcal{G}_{\text{full}}$, which is $\mathcal{G}_{\text{train}}$ plus all of the test edges from the base dataset. Then we remove any queries from the test set that also appeared in the training set. The statistics for the path query datasets are presented in Table 1.

5 Main results

We evaluate the models derived in Section 3 on two tasks: path query answering and knowledge base completion. On both tasks, we show that the compositional training strategy proposed in Section 3.3 leads to substantial performance gains over standard single-edge training. We also compare directly against the KBC results of Socher et al. (2013), demonstrating that previously inferior models now match or outperform state-of-the-art models after compositional training.

Evaluation metric. Numerous metrics have been used to evaluate knowledge base queries, including hits at 10 (percentage of correct answers ranked in the top 10) and mean rank. We evaluate on hits at 10, as well as a normalized version of mean rank, *mean quantile*, which better accounts for the total number of candidates. For a query q, the quantile of a correct answer t is the fraction of incorrect answers ranked after t:

$$\frac{|\{t' \in \mathcal{N}(q) : \text{score}(q, t') < \text{score}(q, t)\}|}{|\mathcal{N}(q)|} \quad (13)$$

The quantile ranges from 0 to 1, with 1 being optimal. Mean quantile is then defined to be the average quantile score over all examples in the dataset. To illustrate why normalization is important, consider a set of queries on the relation `gender`. A model that predicts the incorrect gender on every query would receive a mean rank of 2 (since there are only 2 candidate answers), which is fairly good in absolute terms, whereas the mean quantile would be 0, rightfully penalizing the model.

As a final note, several of the queries in the Freebase path dataset are "type-match trivial" in the sense that all of the type matching candidates $\mathcal{C}(q)$ are correct answers to the query. In this case, mean quantile is undefined and we exclude such queries from evaluation.

Overview. The upper half of Table 2 shows that compositional training improves path querying performance across all models and metrics on both datasets, reducing error by up to 76.2%.

The lower half of Table 2 shows that surprisingly, compositional training also improves performance on knowledge base completion across almost all models, metrics and datasets. On WordNet, TransE benefits the most, with a 43.3% reduction in error. On Freebase, Bilinear benefits the most, with a 38.8% reduction in error.

In terms of mean quantile, the best overall model is TransE (COMP). In terms of hits at 10, the best model on WordNet is Bilinear (COMP), while the best model on Freebase is TransE (COMP).

Deduction and Induction. Table 3 takes a deeper look at performance on path query answering. We divided path queries into two subsets: *deduction* and *induction*. The *deduction* subset consists of queries $q = s/p$ where the source and target entities $[\![q]\!]$ are connected via relations p in the training graph $\mathcal{G}_{\text{train}}$, but the specific query q was never seen during training. Such queries can be answered by performing explicit traversal on the training graph, so this subset tests a model's ability to approximate the underlying training graph and predict the existence of a path from a collection of single edges. The *induction* subset consists of all other queries. This means that at least one edge was missing on all paths following p from source to target in the training graph. Hence, this subset tests a model's generalization ability and its robustness to missing edges.

Performance on the *deduction* subset of the dataset is disappointingly low for models trained with single-edge training: they struggle to answer path queries even when *all edges in the path query have been seen at training time.* Compositional training dramatically reduces these errors, sometimes doubling mean quantile. In Section 6, we analyze how this might be possible. After compositional training, performance on the harder *induction* subset is also much stronger. Even when edges are missing along a path, the models are able to infer them.

Interpretable queries. Although our path datasets consists of random queries, both datasets contain a large number of useful, interpretable queries. Results on a few illustrative examples are shown in Table 4.

		Bilinear			Bilinear-Diag			TransE		
Path query task		SINGLE	COMP	(%red)	SINGLE	COMP	(%red)	SINGLE	COMP	(%red)
WordNet	MQ	84.7	89.4	**30.7**	59.7	90.4	**76.2**	83.7	93.3	**58.9**
	H@10	43.6	54.3	**19.0**	7.9	31.1	**25.4**	13.8	43.5	**34.5**
Freebase	MQ	58.0	83.5	**60.7**	57.9	84.8	**63.9**	86.2	88	**13.0**
	H@10	25.9	42.1	**21.9**	23.1	38.6	**20.2**	45.4	50.5	**9.3**
KBC task		SINGLE	COMP	(%red)	SINGLE	COMP	(%red)	SINGLE	COMP	(%red)
WordNet	MQ	76.1	82.0	**24.7**	76.5	84.3	**33.2**	75.5	86.1	**43.3**
	H@10	19.2	27.3	**10.0**	12.9	14.4	**1.72**	4.6	16.5	**12.5**
Freebase	MQ	85.3	91.0	**38.8**	84.6	89.1	**29.2**	92.7	92.8	**1.37**
	H@10	70.2	76.4	**20.8**	63.2	67.0	**10.3**	78.8	78.6	-0.9

Table 2: **Path query answering and knowledge base completion.** We compare the performance of single-edge training (SINGLE) vs compositional training (COMP). MQ: mean quantile, H@10: hits at 10, %red: percentage reduction in error.

Interpretable Queries	Bilinear SINGLE	Bilinear COMP
`X/institution/institution`$^{-1}$`/profession`	50.0	**93.6**
`X/parents/religion`	81.9	**97.1**
`X/nationality/nationality`$^{-1}$`/ethnicity`$^{-1}$	68.0	**87.0**
`X/has_part/has_instance`$^{-1}$	92.6	**95.1**
`X/type_of/type_of/type_of`	72.8	**79.4**

Table 4: Path query performance (mean quantile) on a selection of interpretable queries. We compare Bilinear SINGLE and Bilinear COMP. Meanings of each query (descending): "What professions are there at X's institution?"; "What is the religion of X's parents?"; "What are the ethnicities of people from the same country as X?"; "What types of parts does X have?"; and the transitive "What is X a type of?". (Note that a relation r and its inverse r^{-1} do not necessarily cancel out if r is not a one-to-one mapping. For example, `X/institution/institution`$^{-1}$ denotes the set of all people who work at the institution X works at, which is not just X.)

Path query task		**WordNet**		**Freebase**	
		Ded.	Ind.	Ded.	Ind.
Bilinear	SINGLE	96.9	66.0	49.3	49.4
	COMP	**98.9**	**75.6**	**82.1**	**70.6**
Bi-Diag	SINGLE	56.3	51.6	49.3	50.2
	COMP	**98.5**	**78.2**	**84.5**	**72.8**
TransE	SINGLE	92.6	71.7	85.3	72.4
	COMP	**99.0**	**87.4**	**87.5**	**76.3**

Table 3: **Deduction and induction.** We compare mean quantile performance of single-edge training (SINGLE) vs compositional training (COMP). Length 1 queries are excluded.

Comparison with Socher et al. (2013). Here, we measure performance on the KBC task in terms of the accuracy metric of Socher et al. (2013). This evaluation involves sampled negatives, and is hence noisier than mean quantile, but makes our results directly comparable to Socher et al. (2013). Our results show that previously inferior models such as the bilinear model can outperform state-of-the-art models after compositional training.

Socher et al. (2013) proposed parametrizing each entity vector as the average of vectors of words in the entity ($w_{\texttt{tad\_lincoln}} = \frac{1}{2}(w_{\texttt{tad}} + w_{\texttt{lincoln}})$, and pretraining these word vectors using the method of Turian et al. (2010). Table 5 reports results when using this approach in conjunction with compositional training. We initialized all models with word vectors from Pennington et al. (2014). We found that compositionally trained models outperform the neural tensor network (NTN) on WordNet, while being only slightly behind on Freebase. (We did not use word vectors in any of our other experiments.)

When the strategy of averaging word vectors to form entity vectors is not applied, our compositional models are significantly better on WordNet and slightly better on Freebase. It is worth noting that in many domains, entity names are not lexically meaningful, so word vector averaging is not

Accuracy	WordNet		Freebase	
	EV	WV	EV	WV
NTN	70.6	86.2	87.2	**90.0**
Bilinear COMP	77.6	**87.6**	86.1	89.4
TransE COMP	**80.3**	84.9	**87.6**	89.6

Table 5: Model performance in terms of accuracy. EV: entity vectors are separate (initialized randomly); WV: entity vectors are average of word vectors (initialized with pretrained word vectors).

always meaningful.

6 Analysis

In this section, we try to understand why compositional training is effective. For concreteness, everything is described in terms of the bilinear model. We will refer to the compositionally trained model as COMP, and the model trained with single-edge training as SINGLE.

6.1 Why does compositional training improve path query answering?

It is tempting to think that if SINGLE has accurately modeled individual edges in a graph, it should accurately model the paths that result from those edges. This intuition turns out to be incorrect, as revealed by SINGLE's relatively weak performance on the path query dataset. We hypothesize that this is due to cascading errors along the path. For a given edge (s, r, t) on the path, single-edge training encourages x_t to be closer to $x_s^\top W_r$ than any other incorrect $x_{t'}$. However, once this is achieved by a margin of 1, it does not push x_t any closer to $x_s^\top W_r$. The remaining discrepancy is noise which gets added at each step of path traversal. This is illustrated schematically in Figure 2.

To observe this phenomenon empirically, we examine how well a model handles each *intermediate* step of a path query. We can do this by measuring the *reconstruction quality* (RQ) of the set vector produced after each traversal operation. Since each intermediate stage is itself a valid path query, we define RQ to be the average quantile over all entities that belong in $[\![q]\!]$:

$$\text{RQ}(q) = \frac{1}{|[\![q]\!]|} \sum_{t \in [\![q]\!]} \text{quantile}(q, t) \qquad (14)$$

When all entities in $[\![q]\!]$ are ranked above all incorrect entities, RQ is 1. In Figure 3, we illustrate how RQ changes over the course of a query.

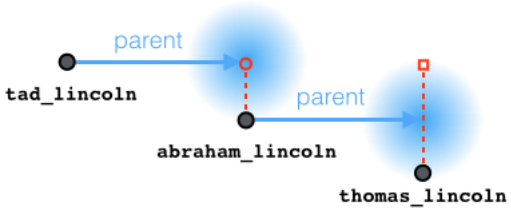

Figure 2: **Cascading errors visualized for TransE.** Each node represents the position of an entity in vector space. The relation `parent` is ideally a simple horizontal translation, but each traversal introduces noise. The red circle is where we expect Tad's parent to be. The red square is where we expect Tad's grandparent to be. Dotted red lines show that error grows larger as we traverse farther away from Tad. Compositional training pulls the entity vectors closer to the ideal arrangement.

Given the nature of cascading errors, it might seem reasonable to address the problem by adding a term to our objective which explicitly encourages $x_s^\top W_r$ to be as close as possible to x_t. With this motivation, we tried adding $\lambda \|x_s^\top W_r - x_t\|_2^2$ term to the objective of the bilinear model and a $\lambda \|x_s + w_r - x_t\|_2^2$ term to the objective of TransE. We experimented with different settings of λ over the range $[0.001, 100]$. In no case did this additional ℓ_2 term improve SINGLE's performance on the path query or single edge dataset. These results suggest that compositional training is a more effective way to combat cascading errors.

6.2 Why does compositional training improve knowledge base completion?

Table 2 reveals that COMP also performs better on the single-edge task of knowledge base completion. This is somewhat surprising, since SINGLE is trained on a training set which distributionally matches the test set, whereas COMP is not. However, COMP's better performance on path queries suggests that there must be another factor at play. At a high level, training on paths must be providing some form of structural regularization which reduces cascading errors. Indeed, paths in a knowledge graph have proven to be important features for predicting the existence of single edges (Lao et al., 2011; Neelakantan et al., 2015). For example, consider the following Horn clause:

$$\text{parents}(x, y) \wedge \text{location}(y, z) \Rightarrow \text{place\_of\_birth}(x, z),$$

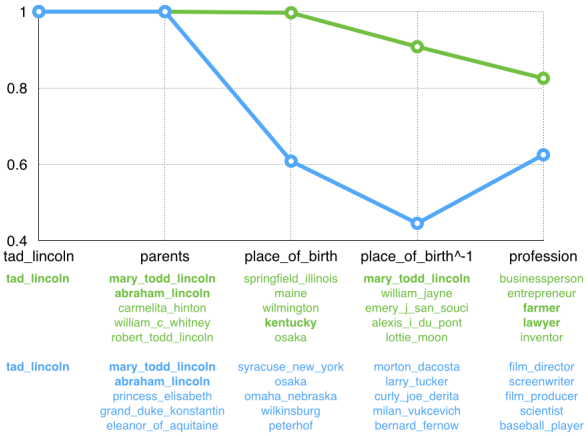

Figure 3: Reconstruction quality (RQ) at each step of the query `tad_lincoln/parents/place_of_birth/place_of_birth`$^{-1}$`/profession`. COMP experiences significantly less degradation in RQ as path length increases. Correspondingly, the set of 5 highest scoring entities computed at each step using COMP (green) is significiantly more accurate than the set given by SINGLE (blue). Correct entities are bolded.

which states that if x has a parent with location z, then x has place of birth z. The body of the Horn clause expresses a path from x to z. If COMP models the path better, then it should be better able to use that knowledge to infer the head of the Horn clause.

More generally, consider Horn clauses of the form $p \Rightarrow r$, where $p = r_1/\ldots/r_k$ is a path type and r is the relation being predicted. Let us focus on Horn clauses with high precision as defined by:

$$\text{prec}(p) = \frac{|[\![p]\!] \cap [\![r]\!]|}{|[\![p]\!]|}, \tag{15}$$

where $[\![p]\!]$ is the set of entity pairs connected by p, and similarly for $[\![r]\!]$.

Intuitively, one way for the model to implicitly learn and exploit such a Horn clause would be to satisfy the following two criteria:

1. The model should ensure a *consistent* spatial relationship between entity pairs that are related by the path type p; that is, keeping $x_s^\top W_{r_1} \ldots W_{r_k}$ close to x_t for *all* valid (s,t) pairs.

2. The model's representation of the path type p and relation r should capture that spatial relationship; that is, $x_s^\top W_{r_1} \ldots W_{r_k} \approx x_t$ implies $x_s^\top W_r \approx x_t$, or simply $W_{r_1} \ldots W_{r_k} \approx W_r$.

We have already seen empirically that SINGLE does not meet criterion 1, because cascading errors cause it to put incorrect entity vectors $x_{t'}$ closer to $x_s^\top W_{r_1} \ldots W_{r_k}$ than the correct entity. COMP mitigates these errors.

To empirically verify that COMP also does a better job of meeting criterion 2, we perform the following: for a path type p and relation r, define $\text{dist}(p,r)$ to be the angle between their corresponding matrices (treated as vectors in $\mathbb{R}^{d^2}$). This is a natural measure because $x_s^\top W_r x_t$ computes the matrix inner product between W_r and $x_s x_t^\top$. Hence, any matrix with small distance from W_r will produce nearly the same scores as W_r for the same entity pairs.

If COMP is better at capturing the correlation between p and r, then we would expect that when $\text{prec}(p)$ is high, compositional training should shrink $\text{dist}(p,r)$ more. To confirm this hypothesis, we enumerated over all 676 possible paths of length 2 (including inverted relations), and examined the proportional reduction in $\text{dist}(p,r)$ caused by compositional training,

$$\Delta\text{dist}(p,r) = \frac{\text{dist}_{\text{COMP}}(p,r) - \text{dist}_{\text{SINGLE}}(p,r)}{\text{dist}_{\text{SINGLE}}(p,r)}. \tag{16}$$

Figure 4 shows that higher precision paths indeed correspond to larger reductions in $\text{dist}(p,r)$.

7 Related work

Knowledge base completion with vector space models. Many models have been proposed for knowledge base completion, including those reviewed in Section 3.4 (Nickel et al., 2011; Bordes et al., 2013; Yang et al., 2015; Socher et al., 2013). Dong et al. (2014) demonstrated that KBC models can improve the quality of relation extraction by serving as graph-based priors. Riedel et al. (2013) showed that such models can be also be directly used for open-domain relation extraction. Our compositional training technique is an orthogonal improvement that could help any composable model.

Distributional compositional semantics. Previous works have explored compositional vector space representations in the context of logic and sentence interpretation. In Socher et al. (2012), a matrix is associated with each word of a sentence, and can be used to recursively modify the meaning of nearby constituents. Grefenstette (2013) ex-

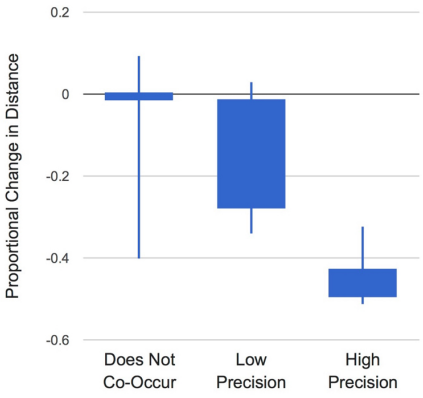

Figure 4: We divide paths of length 2 into high precision (> 0.3), low precision (≤ 0.3), and not co-occuring with r. Here $r =$ `nationality`. Each box plot shows the min, max, and first and third quartiles of $\Delta\text{dist}(p, r)$. As hypothesized, compositional training results in large decreases in $\text{dist}(p, r)$ for high precision paths p, modest decreases for low precision paths, and little to no decreases for irrelevant paths.

plored the ability of tensors to simulate logical calculi. Bowman et al. (2014) showed that recursive neural networks can learn to distinguish important semantic relations. Socher et al. (2014) found that compositional models were powerful enough to describe and retrieve images.

We demonstrate that compositional representations are also useful in the context of knowledge base querying and completion. In the aforementioned work, compositional models produce vectors which represent truth values, sentiment or image features. In our approach, vectors represent sets of entities constituting the denotation of a knowledge base query.

Path modeling. Numerous methods have been proposed to leverage path information for knowledge base completion and question answering. Nickel et al. (2014) proposed combining low-rank models with sparse path features. Lao and Cohen (2010) used random walks as features and Gardner et al. (2014) extended this approach by using vector space similarity to govern random walk probabilities. Neelakantan et al. (2015) addressed the problem of path sparsity by embedding paths using a recurrent neural network. Perozzi et al. (2014) sampled random walks on social networks as training examples, with a different goal to classify nodes in the network. Bordes et al. (2014) embed paths as a sum of relation vectors for question answering. Our approach is unique in modeling the denotation of each intermediate step of a path query, and using this information to regularize the spatial arrangement of entity vectors.

8 Discussion

We introduced the task of answering path queries on an incomplete knowledge base, and presented a general technique for compositionalizing a broad class of vector space models. Our experiments show that compositional training leads to state-of-the-art performance on both path query answering and knowledge base completion.

There are several key ideas from this paper: regularization by augmenting the dataset with paths, representing sets as low-dimensional vectors in a context-sensitive way, and performing function composition using vectors. We believe these three could all have greater applicability in the development of vector space models for knowledge representation and inference.

Reproducibility Our code, data, and experiments are available on the CodaLab platform at `https://www.codalab.org/worksheets/0xfcace41fdeec45f3bc6ddf31107b829f`.

Acknowledgments We would like to thank Gabor Angeli for fruitful discussions and the anonymous reviewers for their valuable feedback. We gratefully acknowledge the support of the Google Natural Language Understanding Focused Program and the National Science Foundation Graduate Research Fellowship under Grant No. DGE-114747.

References

F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In

International Conference on Management of Data (SIGMOD), pages 1247–1250.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2787–2795.

A. Bordes, S. Chopra, and J. Weston. 2014. Question answering with subgraph embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

S. R. Bowman, C. Potts, and C. D. Manning. 2014. Can recursive neural tensor networks learn logical reasoning? In *International Conference on Learning Representations (ICLR)*.

X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 601–610.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

M. Gardner, P. Talukdar, J. Krishnamurthy, and T. Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*.

E. Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. *arXiv preprint arXiv:1304.5823*.

N. Lao and W. W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

N. Lao, T. Mitchell, and W. W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 529–539.

B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *North American Association for Computational Linguistics (NAACL)*, pages 777–782.

A. Neelakantan, B. Roth, and A. McCallum. 2015. Compositional vector space models for knowledge base completion. In *Association for Computational Linguistics (ACL)*.

M. Nickel, V. Tresp, and H. Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning (ICML)*, pages 809–816.

M. Nickel, V. Tresp, and H. Kriegel. 2012. Factorizing YAGO. In *World Wide Web (WWW)*.

M. Nickel, X. Jiang, and V. Tresp. 2014. Reducing the rank in relational factorization models by including observable patterns. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1179–1187.

J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

B. Perozzi, R. Al-Rfou, and S. Skiena. 2014. Deepwalk: Online learning of social representations. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 701–710.

S. Riedel, L. Yao, and A. McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *North American Association for Computational Linguistics (NAACL)*.

R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 1201–1211.

R. Socher, D. Chen, C. D. Manning, and A. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems (NIPS)*, pages 926–934.

R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics (TACL)*, 2:207–218.

J. Turian, L. Ratinov, and Y. Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394.

J. D. Ullman. 1985. Implementation of logical query languages for databases. *ACM Transactions on Database Systems (TODS)*, 10(3):289–321.

B. Yang, W. Yih, X. He, J. Gao, and L. Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.