Compositional Learning of Embeddings for Relation Paths in Knowledge Bases and Text

Kristina Toutanova
Microsoft Research
Redmond, WA, USA

Xi Victoria Lin[*]
Computer Science & Engineering
University of Washington

Wen-tau Yih
Microsoft Research
Redmond, WA, USA

Hoifung Poon
Microsoft Research
Redmond, WA, USA

Chris Quirk
Microsoft Research
Redmond, WA, USA

Abstract

Modeling relation paths has offered significant gains in embedding models for knowledge base (KB) completion. However, enumerating paths between two entities is very expensive, and existing approaches typically resort to approximation with a sampled subset. This problem is particularly acute when text is jointly modeled with KB relations and used to provide direct evidence for facts mentioned in it. In this paper, we propose the first exact dynamic programming algorithm which enables efficient incorporation of all relation paths of bounded length, while modeling both relation types and intermediate nodes in the compositional path representations. We conduct a theoretical analysis of the efficiency gain from the approach. Experiments on two datasets show that it addresses representational limitations in prior approaches and improves accuracy in KB completion.

1 Introduction

Intelligent applications benefit from structured knowledge about the entities and relations in their domains. For example, large-scale knowledge bases (KB), such as Freebase (Bollacker et al., 2008) or DBPedia (Auer et al., 2007), have proven to be important resources for supporting open-domain question answering (Berant et al., 2013; Sun et al., 2015; Yih et al., 2015). In biomedicine, KBs such as the Pathway Interaction Database (NCI-PID) (Schaefer et al., 2009) are crucial for understanding complex diseases such as cancer and for advancing precision medicine. While these knowledge bases are often carefully curated, they are far from complete. In non-static domains, new facts become true or are discovered at a fast pace, making the manual expansion of knowledge bases impractical. Extracting relations from a text corpus (Mintz et al., 2009; Surdeanu et al., 2012; Poon et al., 2015) or inferring facts from the relationships among known entities (Lao and Cohen, 2010) are thus important approaches for populating existing knowledge bases.

Originally proposed as an alternative statistical relational learning method, the *knowledge base embedding* approach has gained a significant amount of attention, due to its simple prediction time computation and strong empirical performance (Nickel et al., 2011; Chang et al., 2014). In this framework, entities and relations in a knowledge base are represented in a continuous space, such as vectors and matrices. Whether two entities have a previously unknown relationship can be predicted by simple functions of their corresponding vectors or matrices. Early work in this direction focuses on exploring various kinds of learning objectives and frameworks, but the model is learned solely from known *direct* relationships between two entities (e.g., `father(barack, sasha)`) (Nickel et al., 2011; Socher et al., 2013; Bordes et al., 2013; Chang et al., 2014; Yang et al., 2015). In contrast, using *multi-step* relation paths (e.g., `husband(barack, `*`michelle`*`)` $\wedge$ `mother(`*`michelle`*`, sasha)` to train KB embeddings has been proposed very recently (Guu et al., 2015; Garcia-Duran et al., 2015; Lin et al., 2015; Neelakantan et al., 2015).

While using relation paths improves model performance, it also poses a critical technical challenge. As the number of possible relation paths between pairs of entities grows exponentially with path length, the training complexity increases sharply. Consequently, existing methods need

[*]This research was conducted during the author's internship at Microsoft Research.

to make approximations by sampling or pruning. The problem is worsened when the input is augmented with unlabeled text, which has been shown to improve performance (Lao et al., 2012; Gardner et al., 2013; Riedel et al., 2013; Gardner et al., 2014; Toutanova and Chen, 2015). Moreover, none of the prior methods distinguish relation paths that differ in the intermediate nodes they pass through (e.g., `michelle` in our example); all represent paths as a sequence of relation types.

In this work, we aim to develop a KB completion model that can incorporate relation paths efficiently. We start from analyzing the procedures in existing approaches, focusing on their time and space complexity. Based on the observation that compositional representations of relation paths are in fact decomposable, we propose a novel dynamic programming method that enables efficient modeling of *all* possible relation paths, while also representing *both* relation types and nodes on the paths.

We evaluated our approach on two datasets. The first is from the domain of gene regulatory networks. Apart from its obvious significance in biomedicine, it offers an excellent testbed for learning joint embedding of KBs and text, as it features existing knowledge bases such as NCI-PID and an even larger body of text that grows rapidly (over one million new articles per year). By modeling intermediate nodes on relation paths, we improve the model by 3 points in mean average precision compared to previous work, while also providing a more efficient algorithm. The second dataset is based on a network derived from WordNet and previously used in work on knowledge base completion. On that dataset we demonstrate the ability of the model to effectively handle longer relation paths composed of a larger set of knowledge base relation types, with smaller positive impact of modeling intermediate nodes.

2 Preliminaries

In this section, we first give a brief overview of the knowledge base and text representation used in this work. We then describe the task of knowledge base completion more formally and introduce our basic model setting and training objective.

Knowledge Base A knowledge base (KB) is represented as a collection of subject-predicate-object triples (s, r, t), where s and t are the subject and object entities from a set $\mathcal{E}$, and r is the predicate from a set $\mathcal{R}$ that denotes a relation-

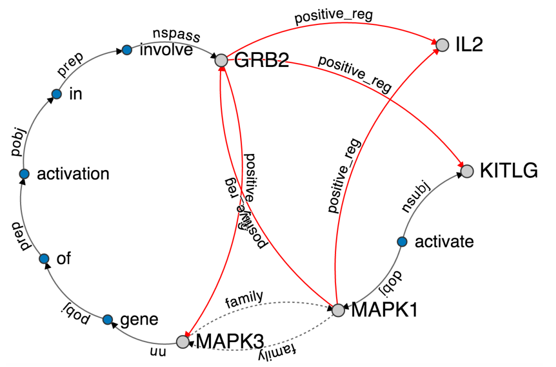

Figure 1: A snapshot depicting the knowledge graph of a gene regulatory network, augmented with gene family and dependency path relations.

ship between s and t. For example, in a KB of movie facts, we may find a triple (`Han_Solo`, `character`, `Star_Wars`), indicating that "*Han Solo is a character in the Star Wars movie.*"

Let $(s, \pi, t) = (s, r_1, e_1, r_2, e_2 \ldots, e_{n-1}, r_n, t)$ be a path in $\mathcal{G}$ with s/t as the start/end entities, $r_1, \ldots, r_n$ as the relation edges and $e_1, \ldots, e_{n-1}$ as the intermediate entities.

In the domain of gene regulations, entities are genes and directed edges represent regulations. At the abstract level, there are two key relations, positive_reg and negative_reg, which signify that the subject gene increases or decreases the activity of the object gene, respectively. Figure 1 shows a snapshot of a gene regulatory network. Genes such as `GRB2` and `MAPK3` are denoted by the light grey nodes. Regulations such as positive_reg are denoted by long edges pointing from subject to object. In addition, some genes stem from a common evolutionary origin and share similar functions. They form a "gene family", such as the `MAPK` family that includes `MAPK1` and `MAPK3`.

To jointly embed text, we use sentences containing co-occurring gene pairs, such as "*GRB2 was involved in the activation of gene MAPK3...*", and augment the above knowledge graph with dependency paths between the gene mentions, following the general approach of Riedel et al. (2013).

Task and Scoring Models The KB completion task is to predict the existence of links (s, r, t) that are not seen in the training knowledge base. More specifically, we focus on ranking object and subject entities for given queries $(s, r, *)$ and $(*, r, t)$.

The basic KB embedding models learn latent vector/matrix representations for entities and relations, and score each possible triple using learned parameters θ and a scoring function $f(s, r, t|\theta)$.

Below, we describe two basic model variations which we build on in the rest of the paper.

BILINEAR The BILINEAR model learns a square matrix $W_r \in \mathbb{R}^{d \times d}$ for each relation $r \in \mathcal{R}$ and a vector $x_e \in \mathbb{R}^d$ for each entity $e \in \mathcal{E}$. The scoring function of a relation triple (s, r, t) is defined as:

$$f(s, r, t|\theta) = x_s^\top W_r x_t. \tag{1}$$

For a knowledge graph $\mathcal{G}$ with $|\mathcal{E}| = N_e$ and $|\mathcal{R}| = N_r$, the parameter size of the BILINEAR model is $O(d^2)$. The large number of parameters make it prone to overfitting, which motivates its diagonal approximation, BILINEAR-DIAG.

BILINEAR-DIAG The BILINEAR-DIAG model restricts the relation representations to the class of diagonal matrices. [1] In this case, the parameter size is reduced to $O(d)$. Although we present model variants in terms of the BILINEAR representation, all experiments in this work are based on the BILINEAR-DIAG special case.

All models are trained using the same loss function, which maximizes the probability of correct subject/object fillers of a given set of triples with relations or relation paths. The probability is defined by a log-linear model normalized over a set of negative samples: $P(t|s, \pi; \theta) = \frac{e^{f(s,\pi,t|\theta)}}{\sum_{t' \in Neg(s,\pi,*) \cup \{t\}} e^{f(s,\pi,t'|\theta)}}$. The probability of subject entities is defined analogously.

Define the loss for a given training triple $L(s, \pi, t|\theta) = -\log P(t|s, \pi; \theta) - \log P(s|t, \pi; \theta)$. The overall loss function is the sum of losses over all triples, with an L_2 regularization term.

$$L(\theta) = \sum_i L(s_i, \pi_i, t_i; \theta) + \lambda ||\theta||^2 \tag{2}$$

3 Relation-path-aware Models

We first review two existing methods using vector space relation paths modeling for KB completion in §3.1. We then introduce our new algorithm that can efficiently take into account all relation paths between two nodes as features and simultaneously model intermediate nodes on relation paths in §3.2. We present a detailed theoretical comparison of the efficiency of these three types of methods in §3.3.

3.1 Prior Approaches

The two approaches we consider here are: using relation paths to generate new auxiliary triples for training (Guu et al., 2015) and using relation paths as features for scoring (Lin et al., 2015).

Both approaches take into account embeddings of relation paths between entities, and both of them used vector space compositions to combine the embeddings of individual relation links r_i into an embedding of the path π. The intermediate nodes e_i are neglected. The natural composition function of a BILINEAR model is matrix multiplication (Guu et al., 2015). For this model, the embedding of a length-n path $\Phi_\pi \in \mathbb{R}^{d \times d}$ is defined as the matrix product of the sequence of relation matrices for the relations in π.

$$\Phi_\pi = W_{r_1} \dots W_{r_n}. \tag{3}$$

For the BILINEAR-DIAG model, all the matrices are diagonal and the computation reduces to coordinate-wise product of vectors in $\mathbb{R}^d$.

3.1.1 Relation Paths as a Compositional Regularizer

In Guu et al. (2015), information from relation paths was used to generate additional auxiliary terms in training, which serve to provide a compositional regularizer for the learned node and relation embeddings. A more limited version of the same method was simultaneously proposed in Garcia-Duran et al. (2015).

The method works as follows: starting from each node in the knowledge base, it samples m random walks of length 2 to a maximum length L, resulting in a list of samples $\{[s_i, \pi_i, t_i]\}$. s_i and t_i are the start and end nodes of the random walk, respectively, and π_i consists of a sequence of intermediate edges and nodes. Each of these samples is used to define a new triple used in the training loss function (eq. 2).

The score of each triple under a BILINEAR composition model is defined as $f(s_i, \pi_i, t_i|\theta) = x_{s_i}^t \Phi_{\pi_i} x_{t_i}$, where Φ_{π_i} is the product of matrices for relation link types in the path (eq. 3).

3.1.2 PRUNED-PATHS: Relation Paths as Compositional Features

Instead of using relation paths to augment the set of training triples, Lin et al. (2015) proposed to

[1] The downside of this approximation is that it enforces symmetry in every relation, i.e., $f(s, r, t) = (x_s \circ x_t)^\top w_r = (x_t \circ x_s)^\top w_r = f(t, r, s)$. However, empirically it has been shown to outperform the Bilinear model when the number of training examples is small (Yang et al., 2015).

use paths (s, π, t) to define the scoring function $f(s, r, t|\theta, \Pi_{s,t})$. Here $\Pi_{s,t}$ denotes the sum of the embeddings of a set of paths π between the two nodes in the graph, weighted by path-constrained random walk probabilities. Their implementation built on the TransE (Bordes et al., 2013) embedding model; in comparison, we formulate a similar model using the BILINEAR model.

We refer to such an approach as the PRUNED-PATHS model, for it discards paths with weights below a certain threshold. We define the model under a BILINEAR composition as follows: Let $\{\pi_1, \pi_2, \ldots, \pi_K\}$ denote a fixed set of path types that can be used as a source of features for the model. We abuse notation slightly to refer to a sequence of types of relation links $r_1, r_2, \ldots, r_n$ in a path in $\mathcal{G}$ as π. We denote by $P(t|s, \pi)$ the path-constrained random walk probability of reaching node t starting from node s and following sequences of relation types as specified in π. We define the weighted path representation $F(s, t)$ for a node pair as the weighted sum of the representations of paths π in the set $\{\pi_1, \pi_2, \ldots, \pi_K\}$ that have non-zero path-constrained random walk probabilities. The representation is defined as: $F(s, t) = \sum_{\pi} w_{|\pi|} P(t|s, \pi)\Phi(\pi)$, where $\Phi(\pi)$ is the path relation type representation from (eq. 3). The weights $w_{|\pi|}$ provide a shared parameter for paths of each length, so that the model may learn to trust the contribution of paths of different lengths differentially. The dependence on θ and the set of paths $\Pi_{s,t}$ between the two nodes in the graph was dropped for simplicity. Figure 2 illustrates the weighted path representation between nodes GRB2 and MAPK3 from Figure 1. [2]

Using the above definition, the score of a candidate triple $f(s, r, t|\theta, \Pi_{s,t})$ is defined as:

$$f(s, r, t) = x_s^\top W_r x_t + \text{vec}(F(s, t))^\top \text{vec}(W_r) \quad (4)$$

The first term of the scoring function is the same as that of the BILINEAR model, and the second term takes into account the similarity of the weighted path representations for (s, t) and the predicted relation r. Here we use element-wise product of the two matrices as the similarity metric. Training and scoring under this model requires explicit constructions of paths, and computing and storing the

[2]Unlike this illustration, the representations of dependency paths are not decomposed into representations of individual edges in our implementation.

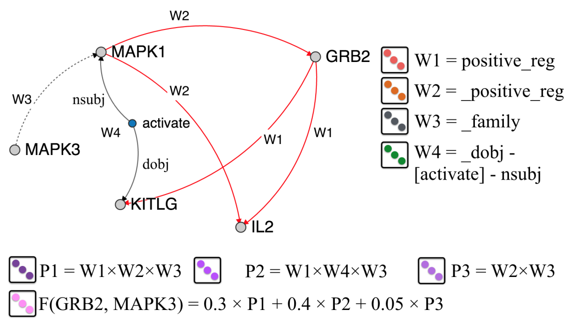

Figure 2: An illustration of the weighted sum of path representations for paths connecting GRB2 and MAPK3 from Figure 1.The prefix "_" of a relation type indicates an inverse relation. The path-constrained random walk probabilities for $P1$, $P2$ and $P3$ are 0.3, 0.4 and 0.05 respectively, and 0.0 for the rest. The path length weights are omitted.

random walk probabilities $P(t|s, \pi)$, which makes it expensive to scale. In the next section, we introduce an algorithm that can efficiently take into account all paths connecting two nodes, and naturally extend it to model the impact of intermediate nodes on the informativeness of the paths.

3.2 ALL-PATHS: A Novel Representation and Algorithm

We now introduce a novel algorithm for efficiently computing and learning the scoring function from (eq. 4), while summing over the set of all paths π up to a certain length L, and additionally modeling the impact of intermediate nodes on the representations $\Phi(\pi)$ of paths. Depending on the characteristics of the knowledge graph and the dimensionality of the learned embedding representations, this method in addition to being more exact, can be faster and take less memory than a method that explicitly generates and prunes full relation paths.

We first define a new representation function for paths of the form $(s, \pi, t) = (s, r_1, e_1, r_2, e_2 \ldots, e_{n-1}, r_n, t)$, which has as a special case the relation-type based function used in prior work, but can additionally model the impact of intermediate nodes e_j on the path representation.

We introduce new parameters w_{e_i} which can impact the representations of paths passing through nodes e_i. w_{e_i} is a scalar weight in our implementation, but vectors of dimensionality d could also be implemented using the same general approach. The embedding of the sample path π under a BILINEAR model is defined as: $\Phi_\pi = W_{r_1} \tanh(w_{e_1}) \cdots W_{r_n} \tanh(w_{e_n})$.

Here the weight of each node is first transformed into the range $[-1, 1]$ using a non-linear tanh function. To derive our exact algorithm, we use quantities $F_l(s,t)$ denoting the weighted sum of path representations for all paths of length l between nodes s and t. The weighted sum of path representations $F(s,t)$ can be written as:

$$\sum_{l=1...L} w_l F_l(s,t), \tag{5}$$

where $F_l(s,t) = \sum_{\pi \in \mathcal{P}_l(s,t)} p(s|t,\pi)\Phi_\pi$ and $P_l(s,t)$ denotes all paths of length l between s, t.

Computing $F(s,t)$ in the naive way by enumerating all possible (s,t) paths is impractical since the number of possible paths can be very large, especially in graphs containing text. Therefore prior work selected a subset of paths through sampling and pruning (Neelakantan et al., 2015; Lin et al., 2015). However, the properties of the BILINEAR composition function for path representation enable us to incrementally build the sums of all path representations exactly, using dynamic programming. Algorithm 1 shows how to compute the necessary quantities $F_l(s,t)$ for all entity pairs in $\mathcal{G}$. [3]

Algorithm 1 Compute the sum of path representations (up to length L) for every entity pair (s,t).

```
Input   : $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, $\{W_r | r \in \mathcal{R}\}$, list of all entity pairs $EP = \{(s,t)\}$.
Output  : $F_L(s,t)$ for every $(s,t) \in EP$.
Initialize:
for $(s,t) \in EP$ do
    if $\mathcal{R}(s,t) \neq \emptyset$ then
        $F_1(s,t) \leftarrow \sum_{r \in \mathcal{R}(s,t)} \tanh(w_t) p(t|s,r) W_{r(s,t)}$
    else
        $F_1(s,t) \leftarrow \mathbf{0}$
    end
end
for $l = 2, \ldots, L$ do
    for $(s,t) \in EP$ do
        $F_l(s,t) \leftarrow \mathbf{0}$
        for $e \in \nu(s)$ do
            $F_l(s,t) \leftarrow F_l(s,t) + F_1(s,e) F_{l-1}(e,t)$
        end
    end
end
```

The key to this solution is that the representations of the longer paths are composed of those of the shorter paths and that the random walk probabilities of relation paths are products of transition probabilities over individual relation edges. The sub-components of paths frequently overlap. For example, all paths π depicted in Figure 2 share the same tail edge family. The algorithms from prior work perform a sum over product representations of paths, but it is more efficient to regroup these into a product of sums. This regrouping allows taking into account exponentially many possible paths without explicitly enumerating them and individually computing their path-constrained random walk probabilities.

After all quantities $F_l(s,t)$ are computed, their weighted sum $F(s,t)$ can be computed using $O(dL)$ operations for each entity pair. These can directly be used to compute the scores of all positive and negative triples for training, using (eq. 4). As we can see, no explicit enumeration or storage of multi-step relation paths between pairs of nodes is needed. To compute gradients of the loss function with respect to individual model parameters, we use a similar algorithm to compute the error for each intermediate edge (e_i, r', e_j) for each group of length l paths between s and t. The algorithm is a variant of forward-backward, corresponding to the forward Algorithm 1.

Notice that directly adding node representations for the paths in the PRUNED-PATHS approach would be infeasible since it would lead to an exposition in the possible path types and therefore the memory and running time requirements of the model. Thus the use of an adaptation of our dynamic programming algorithm to the task of summing over node sequences for a given path type would become necessary to augment the parametric family of the PRUNED-PATHS approach with node representations.

3.3 Efficiency Analysis

We perform a worst-case analysis of the time and memory required for training and evaluation for each of the introduced methods. For the models taking into account relation paths, we assume that all paths up to a certain length L will be modeled[4]. For the basic model text is not taken into account in training, since modeling text and KB relations uniformly in this model did not help performance as seen in the Experiments section. We only take text into account as a source of relation path features or auxiliary path facts.

Notation Let N_e denote the number of nodes in the knowledge graph, E_{kb} the number of knowl-

[3]We further speed up the computation by pre-computing, for each node s, the set of notes t reachable via paths of length l. Then we iterate over non-zero entries only, instead of all pairs in the loops.

[4]If only a subset of paths is used, the complexity of the methods other than ALL-PATHS will be reduced. However, in order to compare the methods in a similar setting, we analyze performance in the case of modeling all multi-step paths.

edge graph links/triples, E_{txt} the number of textual links/triples, a the average number of outgoing links for a node in the graph given the outgoing relation type, N_r the number of distinct relations in the graph, η the number of negative triples for each training triple, and d the dimensionality of entity embeddings and (diagonal) matrix representations of relations.

BILINEAR-DIAG In this basic model, the training time is $\mathrm{O}\big(2d(\eta+1)E_{kb}\big)$ and memory is $\mathrm{O}\big(dN_e + dN_r\big)$. The memory required is for storing embeddings of entities and relations, and the time is for scoring $2(\eta+1)$ triples for every fact in the training KB.

Guu et al. (2015) This method generates training path triples $\{(x,\pi,y)\}$ for all entity pairs connected by paths π up to length L. The number of such triples[5] is $\mathcal{T} = E_{kb} + E_{txt} + \sum_{l=2...L} {N_r}^l \times N_e \times a^l$. The memory required is the memory to store all triples and the set of relation paths, which is $\mathrm{O}\big(\mathcal{T} + \sum_{l=2...L} l{N_r}^l\big)$. The time required includes the time to compute the scores and gradients for all triples and their negative examples (for subject and object position), as well as to compute the compositional path representations of all path types. The time to compute compositional path representations is $\mathrm{O}\big(d\sum_{l=2...L} l{N_r}^l\big)$ and the time spent per triple is $2d(\eta+1)$ as in the basic model. Therefore the overall time per iteration is $\mathrm{O}\big(2d(\eta+1)\mathcal{T}\big)$+$\mathrm{O}\big(d\sum_{l=2...L} l{N_r}^l\big)$. The test time and memory requirements of this method are the same as these of BILINEAR-DIAG, which is a substantial advantage over other methods, if evaluation-time efficiency is important.

PRUNED-PATHS This method computes and stores the values of the random walk probabilities for all pairs of nodes and relation paths, for which these probabilities are non-zero. This can be done in time $\mathrm{O}\big(\mathcal{T}\big)$ where Triples is the same quantity used in the analysis of Guu et al. (2015). The memory requirements of this method are the same as these of (Guu et al., 2015), up to a constant to store random-walk probabilities for paths.

The time requirements are different, however. At training time, we compute scores and update gradients for triples corresponding to direct knowledge base edges, whose number is E_{kb}. For each considered triple, however, we need to compute the sum of representations of path features that are active for the triple. We estimate the average number of active paths per node pair as $\frac{\mathcal{T}}{{N_e}^2}$. Therefore the overall time for this method per training iteration is $\mathrm{O}\big(2d(\eta+1)E_{kb}\frac{\mathcal{T}}{{N_e}^2}\big)$+$\mathrm{O}\big(d\sum_{l=2...L} l{N_r}^l\big)$.

We should note that whether this method or the one of Guu et al. (2015) will be faster in training depends on whether the average number of paths per node pair multiplied by E_{kb} is bigger or smaller than the total number of triples $\mathcal{T}$. Unlike the method of Guu et al. (2015), the evaluation-time memory requirements of this approach are the same as its training memory requirements, or they could be reduced slightly to match the evaluation-time memory requirements of ALL-PATHS, if these are lower as determined by the specific problem instance.

ALL-PATHS This method does not explicitly construct or store fully constructed paths (s,π,t). Instead, memory and time is determined by the dynamic program in Algorithm 1, as well as the forward-backward algorithm for computation of gradients. The memory required to store path representation sums $F_l(s,t)$ is $\mathrm{O}\big(dL{N_e}^2\big)$ in the worst case. Denote $E = E_{kb} + E_{txt}$. The time to compute these sums is $\mathrm{O}\big(dE(1+\sum_{l=2...L}(l-1)N_e)\big)$. After this computation, the time to compute the scores of training positive and negative triples is $\mathrm{O}\big(d2(\eta+1)E_{kb}L\big)$. The time to increment gradients using each triple considered in training is $\mathrm{O}\big(dEL^2\big)$. The evaluation time memory is reduced relative to training time memory by a factor of L and the evaluation time per triple can also be reduced by a factor of L using precomputation.

Based on this analysis, we computed training time and memory estimates for our NCI+Txt knowledge base. Given the values of the quantities from our knowledge graph and $d = 50$, $\eta = 50$, and maximum path length of 5, the estimated memory for (Guu et al., 2015) and PRUNED-PATHS is 4.0×10^{18} and for ALL-PATHS the memory is 1.9×10^9. The time estimates are 2.4×10^{21}, 2.6×10^{25}, and 7.3×10^{15} for (Guu et al., 2015), PRUNED-PATHS, and ALL-PATHS, respectively.

[5]The computation uses the fact that the number of path type sequences of length l is N_r^l. We use a, the average branching factor of nodes given relation types, to derive the estimated number of triples of a given relation type for a path of length l.

Model	KB		KB and Text	
	MAP	HITS@10	MAP	HITS@10
BILINEAR-DIAG (Guu et al., 2015) d=100	12.48	19.66	12.48	19.66
BILINEAR-DIAG d=100	28.56	39.92	28.56	39.92
BILINEAR-DIAG d=2000	30.16	42.51	30.16	42.51
+Guu et al. (2015) d=100 orig.	23.20	34.84	23.20	34.84
+Guu et al. (2015) d=100 reimpl.	29.13	40.59	30.25	41.45
PRUNED-PATHS d=100 c=1000	32.31	43.16	36.42	48.22
PRUNED-PATHS d=100 c=100	32.31	43.16	36.79	48.27
PRUNED-PATHS d=100 c=1	32.31	43.16	37.03	48.26
ALL-PATHS d=100	32.31	43.16	36.24	48.60
ALL-PATHS+NODES d=100	**33.92**	**45.96**	**39.31**	**52.53**

Table 1: KB completion results on NCI-PID test: comparison of our compositional learning approach (ALL-PATHS+NODES) with baseline systems. d is the embedding dimension; sampled paths occurring less than c times were pruned in PRUNED-PATHS.

4 Experiments

Our experiments are designed to study three research questions: (*i*) What is the impact of using path representations as a source of compositional regularization as in (Guu et al., 2015) versus using them as features for scoring as in PRUNED-PATHS and ALL-PATHS? (*ii*) What is the impact of using textual mentions for KB completion in different models? (*iii*) Does modeling intermediate path nodes improve the accuracy of KB completion?

Datasets We used two datasets for evaluation: NCI-PID and WordNet.

For the first set of experiments, we used the Pathway Interaction Database (NCI-PID) (Schaefer et al., 2009) as our knowledge base, which was created by editors from the Nature Publishing Groups, in collaboration with the National Cancer Institute. It contains a collection of high-quality gene regulatory networks (also referred to as *pathways*). The original networks are in the form of hypergraphs, where nodes could be complex gene products (e.g., "protein complex" with multiple proteins bound together) and regulations could have multiple inputs and outputs. Following the convention of most network modeling approaches, we simplified the hypergraphs into binary regulations between genes (e.g., GRB2 positive_reg MAPK3), which yields a graph with 2774 genes and 14323 triples. The triples are then split into train, dev, and test sets, of size 10224, 1315, 2784, respectively. We identified genes belonging to the same family via the common letter prefix in their names, which adds 1936 triples to training.

As a second dataset, we used a WordNet KB with the same train, dev, and test splits as Guu et al. (2015). There are 38,696 entities and 11 types of knowledge base relations. The KB includes 112,581 triples for training, 2,606 triples for validation, and 10,544 triples for testing. WordNet does not contain textual relations and is used for a more direct comparison with recent works.

Textual Relations We used PubMed abstracts for text for NCI-PID. We used the gene mentions identified by Literome (Poon et al., 2014), and considered sentences with co-occurring gene pairs from NCI-PID. We defined textual relations using the fully lexicalized dependency paths between two gene mentions, as proposed in Riedel et al. (2013). Additionally, we define trigger-mapped dependency paths, where only important "trigger" words are lexicalized and the rest of the words are replaced with a wild-card character X. A set of 333 words often associated with regulation events in Literome (e.g. *induce*, *inhibit*, *reduce*, *suppress*) were used as trigger words. To avoid introducing too much noise, we only included textual relations that occur at least 5 times between mentions of two genes that have a KB relation. This resulted in 3,827 distinct textual relations and 1,244,186 mentions.[6] The number of textual relations is much larger than that of KB relations, and it helped induce much larger connectivity among genes (390,338 pairs of genes are directly connected in text versus 12,100 pairs in KB).

Systems ALL-PATHS denotes our compositional learning approach that sums over all paths using

[6]Modeling such a large number of textual relations introduces sparsity, which necessitates models such as (Toutanova et al., 2015; Verga et al., 2015) to derive composed representations of text. We leave integration with such methods for future work.

dynamic programming; ALL-PATHS+NODES additionally models nodes in the paths. PRUNED-PATHS denotes the traditional approach that learns from sampled paths detailed in §3.1.2; paths with occurrence less than a cutoff are pruned ($c = 1$ in Table 1 means that all sampled paths are used). The most relevant prior approach is Guu et al. (2015). We ran experiments using both their publicly available code and our re-implementation. We also included the BILINEAR-DIAG baseline.

Implementation Details We used batch training with RProp (Riedmiller and Braun, 1993). The L_2 penalty λ was set to 0.1 for all models, and the entity vectors x_e were normalized to unit vectors. For each positive example we sample 500 negative examples. For our implementation of (Guu et al., 2015), we run 5 random walks of each length starting from each node and we found that adding a weight β to the multi-step path triples improves the results. After preliminary experimentation, we fixed β to 0.1. Models using KB and textual relations were initialized from models using KB relations only[7]. Model training was stopped when the development set MAP did not improve for 40 iterations; the parameters with the best MAP on the development set were selected as output. Finally, we used only paths of length up to 3 for NCI-PID and up to length 5 for WordNet.[8]

Evaluation metrics We evaluate our models on their ability to predict the subjects/objects of knowledge base triples in the test set. Since the relationships in the gene regulation network are frequently not one-to-one, we use the mean average precision (MAP) measure instead of the mean reciprocal rank often used in knowledge base completion works in other domains. In addition to MAP, we use Hits@10, which is the percentage of correct arguments ranked among the top 10 predictions[9]. We compute measures for ranking both the object entities $(s, r, *)$ and the subject entities$(*, r, t)$. We report evaluation metrics computed on the union of the two query set.

NCI-PID Results Table 1 summarizes the knowledge base (KB) completion results on the NCI-PID test. The rows compare our compositional learning approach ALL-PATHS+NODES with prior approaches. The comparison of the two columns demonstrates the impact when text is jointly embedded with KB. Our compositional learning approach significantly outperforms all other approaches in both evaluation metrics (MAP and HITS@10). Moreover, jointly embedding text and KB led to substantial improvement, compared to embedding KB only.[10] Finally, modeling nodes in the paths offers significant gains (ALL-PATHS+NODE gains 3 points in MAP over ALL-PATHS), with statistical significance ($p < .001$) according to a McNemar test.

Evaluating the effect of path pruning on the traditional approach (PRUNED-PATHS) is quite illuminating. As the number of KB relations is relatively small in this domain, when only KB is embedded, most paths occur frequently. So there is little difference between the heaviest pruned version (c=1000) and the lightest (c=1). When textual relations are included, the cutoff matters more, although the difference was small as many rarer textual relations were already filtered beforehand. In either case, the accuracy difference between ALL-PATHS and PRUNED-PATHS is small, and ALL-PATHS mainly gains in efficiency. However, when nodes are modeled, the compositional learning approach gains in accuracy as well, especially when text is jointly embedded.

Comparison among the baselines also offers valuable insights. The implementation of Guu et al. (2015) with default parameters performed significantly worse than our re-implementation. Also, our re-implementation achieves only a slight gain over the BILINEAR-DIAG baseline, whereas the original implementation obtains substantial improvement over its own version of BILINEAR-DIAG. These results underscore the importance of hyper-parameters and optimization, and invite future systematic research on the impact of such modeling choices.[11]

[7]In addition, models using path training were initialized from models trained on direct relation links only.

[8]Using paths up to length 4 on NCI-PID did not perform better.

[9]As a common practice in KB completion evaluation, for both MAP and Hits@10, we filtered out the other correct answers when ranking a particular triple to eliminate ranking penalization induced by other correct predictions.

[10]Text did not help for the models in the first four rows in the Table, possibly because in these approaches text and KB information are equally weighted in the loss function and the more numerous textual triples dominate the KB ones.

[11]The differences between our two implementations are: max-margin loss versus softmax loss and stochastic gradient training versus batch training.

Model	MAP	HITS@10
BILINEAR-DIAG (Guu et al., 2015)	N/A	12.9
BILINEAR-DIAG	8.0	12.2
+Guu et al. (2015)	N/A	14.4
PRUNED-PATHS $l = 3$ c=10	9.5	14.8
PRUNED-PATHS $l = 3$ c=1	9.5	14.9
PRUNED-PATHS $l = 5$ c=10	8.9	14.4
ALL-PATHS $l = 3$	9.4	14.7
ALL-PATHS+NODES l=3	9.4	15.2
ALL-PATHS $l = 5$	9.6	16.6
ALL-PATHS+NODES l=5	**9.8**	**16.7**

Table 2: KB completion results on the WordNet test set: comparison of our compositional learning approach (ALL-PATHS) with baseline systems. The maximum length of paths is denoted by l. Sampled paths occurring less than c times were pruned in PRUNED-PATHS.

WordNet Results Table 2 presents a comparative evaluation on the WordNet dataset. This dataset has a larger number of knowledge base relation types compared to NCI-PID, and longer relation paths in this KB are expected to be beneficial. Guu et al. (2015) evaluated their compositional regularization approach on this dataset and we can directly compare to their results. The first two rows in the Table show the baseline BILINEAR-DIAG model results according to the results reported in (Guu et al., 2015) and our implementation. The MAP results were not reported in Guu et al. (2015); hence the NA value for MAP in row one.[12] On this dataset, our implementation of the baseline model does not have substantially different results than Guu et al. (2015) and we use their reported results for the baseline and compositionally trained model.

Compositional training improved performance in Hits@10 from 12.9 to 14.4 in Guu et al. (2015), and we find that using PRUNED-PATHS as features gives similar, but a bit higher performance gains.

The PRUNED-PATHS method is evaluated using count cutoffs of 1 and 10, and maximum path lengths of 3 and 5. As can be seen, lower count cutoff performed better for paths up to length 3, but we could not run the method with path lengths up to 5 and count cutoff of 1, due to excessive memory requirements (more than 248GB). When using count cutoff of 10, paths up to length 5 performed worse than paths up to length 3. This performance degradation could be avoided with a staged training regiment where models with shorter paths are first trained and used to initialize models using longer paths.

The performance of the ALL-PATHS method can be seen for maximum paths up to lengths 3 and 5, and with or without using features on intermediate path nodes.[13] As shown in Table 2, longer paths were useful, and features on intermediate nodes were also beneficial. We tested the significance of the differences between several pairs of models and found that nodes led to significant improvement ($p < .002$) for paths of length up to 3, but not for the setting with longer paths. All models using path features are significantly better than the baseline BILINEAR-DIAG model.

To summarize both sets of experiments, the ALL-PATHS approach allows us to efficiently include information from long KB relation paths as in WordNet, or paths including both text and KB relations as in NCI-PID. Our dynamic programming algorithm considers relation paths efficiently, and is also straightforwardly generalizable to include modeling of intermediate path nodes, which would not be directly possible for the PRUNED-PATHS approach. Using intermediate nodes was beneficial on both datasets, and especially when paths could include textual relations as in the NCI-PID dataset.

5 Conclusions

In this work, we propose the first approach to efficiently incorporate all relation paths of bounded length in a knowledge base, while modeling both relations and intermediate nodes in the compositional path representations. Experimental results on two datasets show that it outperforms prior approaches by modeling intermediate path nodes. In the future, we would like to study the impact of relation paths for additional basic KB embedding models and knowledge domains.

Acknowledgments

The authors would like to thank the anonymous reviewers and meta-reviewer for suggestions on making the paper stronger, as well as Kenton Lee and Aria Haghighi for their helpful comments on the draft.

[12]We ran the trained model distributed by Guu et al. (2015) and obtained a much lower Hits@10 value of 6.4 and MAP of of 3.5. Due to the discrepancy, we report the original results from the authors' paper which lack MAP values instead.

[13]To handle the larger scale of the WordNet dataset in terms of the number of nodes in the knowledge base, we enforced a maximum limit (of 100) on the degree of a node that can be an intermediate node in a multi-step relation path.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.

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, October. Association for Computational Linguistics.

Kurt 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 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

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

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Alberto Garcia-Duran, Antoine Bordes, and Nicolas Usunier. 2015. Composing relationships with translations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 286–290, Lisbon, Portugal, September. Association for Computational Linguistics.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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, September. Association for Computational Linguistics.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. 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, September. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *ACL*.

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 machine learning (ICML-11)*, pages 809–816.

Hoifung Poon, Chris Quirk, Charlie DeZiel, and David Heckerman. 2014. Literome: Pubmed-scale genomic knowledge base in the cloud. *Bioinformatics*, 30(19):2840–2842.

Hoifung Poon, Kristina Toutanova, and Chris Quirk. 2015. Distant supervision for cancer pathway extraction from text. In *PSB*.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Martin Riedmiller and Heinrich Braun. 1993. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE.

Carl F Schaefer, Kira Anthony, Shiva Krupa, Jeffrey Buchoff, Matthew Day, Timo Hannay, and Kenneth H Buetow. 2009. PID: the pathway interaction database. *Nucleic acids research*, 37(suppl 1):D674–D679.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems (NIPS)*.

Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1045–1055. International World Wide Web Conferences Steering Committee.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*.

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. Association for Computational Linguistics.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2015. Multilingual relation extraction using compositional universal schema. *arxiv*, abs/1511.06396.

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 *International Conference on Learning Representations (ICLR)*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. 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 1321–1331, Beijing, China, July. Association for Computational Linguistics.