

CoKE: Contextualized Knowledge Graph Embedding

Quan Wang*, Pingping Huang*, Haifeng Wang, Songtai Dai, Wenbin Jiang
Jing Liu, Yajuan Lyu, Yong Zhu, Hua Wu

Baidu Inc., Beijing, China

{wangquan05, huangpingping, wanghaifeng, daisongtai, jiangwenbin,
liujing46, lvayajuan, zhuyong, wu_hua}@baidu.com

Abstract

Knowledge graph embedding, which projects symbolic entities and relations into continuous vector spaces, is gaining increasing attention. Previous methods allow a single static embedding for each entity or relation, ignoring their intrinsic contextual nature, i.e., entities and relations may appear in different graph contexts, and accordingly, exhibit different properties. This work presents *Contextualized Knowledge Graph Embedding* (CoKE), a novel paradigm that takes into account such contextual nature, and learns dynamic, flexible, and fully contextualized entity and relation embeddings. Two types of graph contexts are studied: edges and paths, both formulated as sequences of entities and relations. CoKE takes a sequence as input and uses a Transformer encoder to obtain contextualized representations. These representations are hence naturally adaptive to the input, capturing contextual meanings of entities and relations therein. Evaluation on a wide variety of public benchmarks verifies the superiority of CoKE in link prediction and path query answering. It performs consistently better than, or at least equally well as current state-of-the-art in almost every case, in particular offering an absolute improvement of 19.7% in H@10 on path query answering. Our code is available at <https://github.com/paddlepaddle/models/tree/develop/PaddleKG/CoKE>.

1 Introduction

Recent years have seen rapid progress in knowledge graph (KG) construction and application. A KG is typically a multi-relational graph composed of entities as nodes and relations as different types of edges. Each edge is represented as a subject-relation-object triple (s, r, o) , indicating a specific relation between the two entities. Although such triples are effective in organizing knowledge, their symbolic nature makes them difficult to handle by

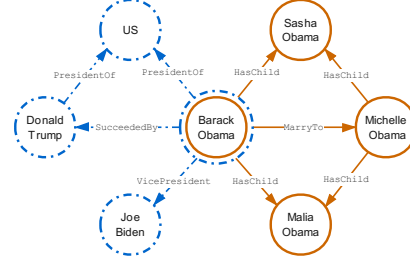


Figure 1: An example of BarackObama, where the left subgraph shows his political role (dashed blue) and the right one his family role (solid orange).

most learning algorithms. KG embedding, which aims to project symbolic entities and relations into continuous vector spaces, has thus been proposed and quickly gained broad attention (Nickel et al., 2016a; Wang et al., 2017). These embeddings preserve the inherent structures of KGs, and have shown to be beneficial in a variety of downstream tasks, e.g., relation extraction (Weston et al., 2013; Riedel et al., 2013) and question answering (Bordes et al., 2014; Yang et al., 2019).

Current approaches typically learn for each entity or relation a single static representation, to describe its global meaning in a given KG. However, entities and relations rarely appear in isolation. Instead, they form rich, varied graph contexts such as edges, paths, or even subgraphs. We argue that entities and relations, when involved in different graph contexts, might exhibit different meanings, just like words do when they appear in different textual contexts (Peters et al., 2018). Figure 1 provides an example of entity BarackObama. The left subgraph (dashed blue) shows his political role as a former president of US, while the right one (solid orange) shows his family role as a husband and a father, which possess quite different properties. Take relation HasPart as another example, which also presents contextualized meanings, e.g., composition-related as (Table, HasPart, Leg) and location-related as (Atlantics, HasPart,

*Equal contribution.

NewYorkBay) (Xiao et al., 2016). Learning entity and relation representations that could effectively capture their contextual meanings poses a new challenge to KG embedding.

Inspired by recent advances in contextualized word embedding (Devlin et al., 2019), we propose *Contextualized Knowledge Graph Embedding* (or CoKE for short), a novel KG embedding paradigm that is flexible, dynamic, and fully contextualized. Unlike previous methods that allow a single static representation for each entity or relation, CoKE models that representation as a function of input graph contexts. Two types of graph contexts are considered: edges and paths, both formalized as sequences of entities and relations. Given an input sequence, CoKE employs a stack of Transformer (Vaswani et al., 2017) blocks to encode the input and obtain contextualized representations for its components. The model is then trained by predicting a missing component in the sequence, based on these contextualized representations. In this way, CoKE learns KG embeddings dynamically adaptive to each input sequence, capturing contextual meanings of entities and relations therein.

We evaluate CoKE with two tasks: link prediction and path query answering (Guu et al., 2015). Both can be formulated exactly in the same way as how CoKE is trained, i.e., to predict a missing entity from a given sequence (triple or path). CoKE performs extremely well in these tasks. It outperforms, or at least performs equally well as current state-of-the-art in almost every case. In particular, it offers an absolute improvement of up to 19.7% in H@10 on path query answering, demonstrating its superior capability for multi-hop reasoning. Though using Transformer, CoKE is still parameter efficient, achieving better or comparable results with much fewer parameters. Visualization further demonstrates that CoKE can discern fine-grained contextual meanings of entities and relations.

We summarize our contributions as follows: (1) We propose the notion of contextualized KG embedding, which differs from previous paradigms by modeling contextual nature of entities and relations in KGs. (2) We devise a new approach CoKE to learn fully contextualized KG embeddings. We show that CoKE can be naturally applied to a variety of tasks like link prediction and path query answering. (3) Extensive experiments demonstrate the superiority of CoKE. It achieves new state-of-the-art results on a number of public benchmarks.

2 Related Work

KG embedding aims at learning distributed representations for entities and relations of a given KG. Recent years have witnessed increasing interest in this task, and various KG embedding techniques have been devised, e.g., translation-based models (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015b), simple semantic matching models (Yang et al., 2015; Nickel et al., 2016b; Trouillon et al., 2016), and neural network models (Dettmers et al., 2018; Jiang et al., 2019; Nguyen et al., 2018). We refer readers to (Nickel et al., 2016a; Wang et al., 2017) for a thorough review. Most of these traditional models learn a static, global representation for each entity or relation, solely from individual subject-relation-object triples.

Beyond triples, recent work tried to use more global graph structures like multi-hop paths (Lin et al., 2015a; Das et al., 2017) and k -degree neighborhoods (Feng et al., 2016; Schlichtkrull et al., 2017) to learn better embeddings. Although such approaches take into account rich graph contexts, they are not “contextualized”, still learning a static global representation for each entity/relation.

The contextual nature of entities and relations has been noted previously, but from distinct views. Consider a classic translation-based model TransE (Bordes et al., 2013). To overcome its disadvantages in dealing with 1-to-N, N-to-1 and N-to-N relations, some researchers introduced relation-specific projections, by which an entity would get different projected representations when involved in different relations (Wang et al., 2014; Lin et al., 2015b; Ji et al., 2015). Xiao et al. (2016) noted that relations can be polysemous, showing different meanings with different entity pairs. So they modeled relations as mixtures of Gaussians to deal with this polysemy issue. Although similar phenomena have been touched upon in previous work, there is little formal discussion about the contextual nature of KGs, and the solutions, of course, are not “contextualized”.

This work is inspired by recent advances in learning contextualized word representations (McCann et al., 2017; Peters et al., 2018; Devlin et al., 2019), by drawing connections of graph edges/paths to natural language phrases/sentences. Such connections have been studied extensively in graph embedding (Perozzi et al., 2014; Grover and Leskovec, 2016; Ristoski and Paulheim, 2016; Cochez et al., 2017). But most of these approaches

obtain static embeddings via traditional word embedding techniques, and fail to capture the contextual nature of entities and relations.

3 Our Approach

Unlike previous methods that assign a single static representation to each entity/relation learned from the whole KG, CoKE models that representation as a function of each individual graph context, i.e., an edge or a path. Given a graph context as input, CoKE employs Transformer blocks to encode the input and obtain contextualized representations for entities and relations therein. The model is trained by predicting a missing entity in the input, based on these contextualized representations. Figure 2 gives an overview of our approach.

3.1 Problem Formulation

We are given a KG composed of subject-relation-object triples $\{(s, r, o)\}$. Each triple indicates a relation $r \in \mathcal{R}$ between two entities $s, o \in \mathcal{E}$, e.g., (BarackObama, HasChild, SashaObama). Here, \mathcal{E} is the entity vocabulary and \mathcal{R} the relation set. These entities and relations form rich, varied graph contexts. Two types of graph contexts are considered here: edges and paths, both formalized as sequences composed of entities and relations.

- An *edge* $s \rightarrow r \rightarrow o$ is a sequence formed by a triple, e.g., BarackObama \rightarrow HasChild \rightarrow SashaObama. This is the basic unit of a KG, and also the simplest form of graph contexts.
- A *path* $s \rightarrow r_1 \rightarrow \dots \rightarrow r_k \rightarrow o$ is a sequence formed by a list of relations linking two entities, e.g., BarackObama \rightarrow HasChild (Sasha) \rightarrow LivesIn (US) \rightarrow OfficialLanguage \rightarrow English.¹ The length of a path is defined as the number of relations therein. The example above is a path of length 3. Edges can be viewed as special paths of length 1.

Here we follow (Guu et al., 2015) and exclude intermediate entities from paths, by which the paths will get a close relationship with Horn clauses and first-order logic rules (Lao and Cohen, 2010). We leave the investigation of other path forms for future work. Given edges and paths that reveal rich graph structures, the aim of CoKE is to learn entity and relation representations dynamically adaptive to each input graph context.

¹Entities in parentheses (Sasha and US) are not components of the path, just used to show how the path is generated.

3.2 Model Architecture

CoKE borrows ideas from recent techniques for learning contextualized word embeddings (Devlin et al., 2019). Given a graph context, i.e., an edge or a path, we unify the input as a sequence $X = (x_1, x_2, \dots, x_n)$, where the first and last elements are entities from \mathcal{E} , and the others in between are relations from \mathcal{R} . For each element x_i in X , we construct its input representation as:

$$\mathbf{h}_i^0 = \mathbf{x}_i^{\text{ele}} + \mathbf{x}_i^{\text{pos}},$$

where $\mathbf{x}_i^{\text{ele}}$ is the element embedding and $\mathbf{x}_i^{\text{pos}}$ the position embedding. The former is used to identify the current element, and the latter its position in the sequence. We allow an element embedding for each entity/relation in $\mathcal{E} \cup \mathcal{R}$, and a position embedding for each position within length K .

After constructing all input representations, we feed them into a stack of L successive Transformer encoders (Vaswani et al., 2017) to encode the sequence and obtain:

$$\mathbf{h}_i^\ell = \text{Transformer}(\mathbf{h}_i^{\ell-1}), \quad \ell = 1, 2, \dots, L,$$

where \mathbf{h}_i^ℓ is the hidden state of x_i after the ℓ -th layer. Unlike sequential left-to-right or right-to-left encoding strategies, Transformer uses a multi-head self-attention mechanism, which allows each element to attend to all elements in the sequence, and thus is more effective in context modeling. As the use of Transformer has become ubiquitous recently, we omit a detailed description of the model architecture and refer readers to (Vaswani et al., 2017). The final hidden states $\{\mathbf{h}_i^L\}_{i=1}^n$ are taken as the desired representations for entities and relations within the specific graph context X . These representations are naturally contextualized, automatically adaptive to the input.

3.3 Model Training

To train the model, we design an entity prediction task, i.e., to predict a missing entity from a given graph context. This task amounts to single-hop or multi-hop question answering on KGs.

- Each *edge* $s \rightarrow r \rightarrow o$ is associated with two training instances: $? \rightarrow r \rightarrow o$ and $s \rightarrow r \rightarrow ?$. It is a single-hop question answering task, e.g., BarackObama \rightarrow HasChild $\rightarrow ?$ is to answer “Who is the child of Barack Obama?”.
- Each *path* $s \rightarrow r_1 \rightarrow \dots \rightarrow r_k \rightarrow o$ is also associated with two training instances, one

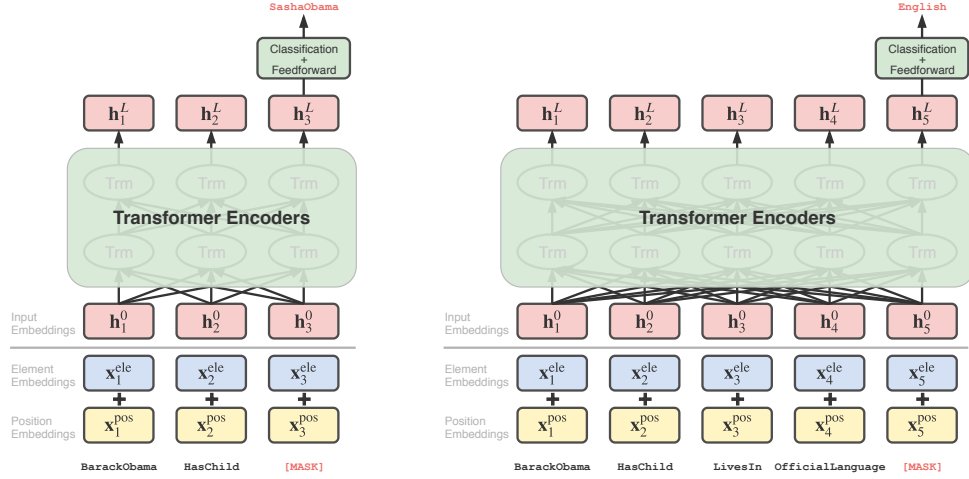


Figure 2: Overall framework of CoKE. An edge (left) or a path (right) is given as an input sequence, with an entity replaced by a special token `[MASK]`. The input is then fed into a stack of Transformer encoder blocks. The final hidden state corresponding to `[MASK]` is used to predict the target entity.

to predict s and the other to predict o . This is a multi-hop question answering task, e.g., $\text{BarackObama} \rightarrow \text{HasChild} \rightarrow \text{LivesIn} \rightarrow \text{OfficialLanguage} \rightarrow ?$ is to answer “*What is the official language of the country where Barack Obama’s child lives in?*”.

This entity prediction task resembles the masked language model (MLM) task studied in (Devlin et al., 2019). But unlike MLM that randomly picks some input tokens to mask and predict, we restrict the masking and prediction solely to entities in a given edge/path, so as to create meaningful question answering instances. Moreover, many downstream tasks considered in the evaluation phase, e.g., link prediction and path query answering, can be formulated exactly in the same way as entity prediction (will be detailed in § 4), which avoids training-test discrepancy.

During training, for each edge or path unified as a sequence $X = (x_1, \dots, x_n)$, we create two training instances, one by replacing x_1 with a special token `[MASK]` (to predict s), and the other by replacing x_n with `[MASK]` (to predict o). Then, the masked sequence is fed into the Transformer encoding blocks. The final hidden state corresponding to `[MASK]`, i.e., \mathbf{h}_1^L or \mathbf{h}_n^L , after a feedforward layer, is used to predict the target entity, via a standard softmax classification layer:

$$\mathbf{z}_1 = \text{Feedforward}(\mathbf{h}_1^L), \mathbf{z}_n = \text{Feedforward}(\mathbf{h}_n^L), \\ \mathbf{p}_1 = \text{softmax}(\mathbf{E}^{\text{ele}} \mathbf{z}_1), \mathbf{p}_n = \text{softmax}(\mathbf{E}^{\text{ele}} \mathbf{z}_n).$$

Here, $\mathbf{z}_1/\mathbf{z}_n$ is the hidden state of $\mathbf{h}_1^L/\mathbf{h}_n^L$ after the feedforward layer, $\mathbf{E}^{\text{ele}} \in \mathbb{R}^{V \times D}$ the classification

weight shared with the input element embedding matrix, D the hidden size, V the entity vocabulary size, and $\mathbf{p}_1/\mathbf{p}_n$ the predicted distribution of x_1/x_n (s/o) over all entities. Figure 2 provides a visual illustration of this whole process.

We use cross-entropy between the label ($\mathbf{y}_1/\mathbf{y}_n$) and the prediction $\mathbf{p}_1/\mathbf{p}_n$ as our training loss:

$$\mathcal{L}(X) = - \sum_t y_t^1 \log p_t^1 - \sum_t y_t^n \log p_t^n,$$

where y_t^1/y_t^n is the t -th component of $\mathbf{y}_1/\mathbf{y}_n$, and p_t^1/p_t^n the t -th component of $\mathbf{p}_1/\mathbf{p}_n$. As a one-hot label here will restrict each entity prediction task to a single correct answer, we use a label smoothing strategy to lessen this restriction, i.e., we set $y_t = \epsilon$ for the target entity, and $y_t = \frac{1-\epsilon}{V-1}$ for each of the other entities.

4 Experiments

We demonstrate the effectiveness of CoKE in link prediction and path query answering. We further visualize CoKE embeddings to show how they can discern contextual usage of entities and relations.

4.1 Link Prediction

This task is to complete a triple (s, r, o) with s or o missing, i.e., to predict $? \rightarrow r \rightarrow o$ or $s \rightarrow r \rightarrow ?$ (Bordes et al., 2013). It is in the same form as our training task, i.e., entity prediction within edges.

Datasets We conduct experiments on four widely used benchmarks. The statistics of the datasets are summarized in Table 1. FB15k and WN18 were

	FB15k	WN18	FB15k-237	WN18RR
Entities	14,951	40,943	14,541	40,943
Relations	1,345	18	237	11
Train	483,142	141,442	272,115	86,835
Dev	50,000	5,000	17,535	3,034
Test	59,071	5,000	20,466	3,134

Table 1: Number of entities, relations, and triples in each split of the four benchmarks.

introduced in (Bordes et al., 2013), with the former sampled from Freebase and the latter from WordNet. FB15k-237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2018) are their modified versions, which exclude inverse relations and are harder to fit.

Training Details In this task, we train our model with only *triples* from the *training* set. The maximum input sequence length is hence restricted to $K = 3$. We use the following configuration for CoKE: the number of Transformer blocks $L = 6$, number of self-attention heads $A = 4$, hidden size $D = 256$, and feed-forward size $2D = 512$. We employ dropout on all layers, with the rate tuned in $\rho \in \{0.1, 0.5\}$. The label smoothing rate is tuned in $\epsilon \in (0, 1]$ with steps of 0.05. We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate $\eta \in \{3e^{-4}, 5e^{-4}\}$. We also use learning rate warmup over the first 10% training steps and linear decay of the learning rate. We train with batch size $B \in \{512, 1024\}$ for at most 1000 epochs. The best hyperparameter setting is selected according to MRR (described later) on the dev set.

Evaluation Protocol During evaluation, given a test triple (s, r, o) , we replace s with $[\text{MASK}]$, feed the sequence into CoKE, and obtain the predicted distribution of s over all entities. We sort the distribution probabilities in descending order and get the rank of s . During ranking, we remove any s' that (s', r, o) already exists in the training, dev, or test set, i.e., a *filtered* setting (Bordes et al., 2013). This whole procedure is repeated while predicting o . We report the mean reciprocal rank (MRR) and the proportion of ranks no larger than n ($H@n$).

Main Results Tables 2 and 3 report link prediction results on the four datasets. We select competitive baselines from the most recent publications with good results reported. Our baselines are categorized into two groups: methods that use triples alone and methods that further integrate rich graph contexts or logic rules (rules have a close relation-

ship to multi-hop paths). CoKE falls into the first group as it uses only triples from the training set.

The results are quite promising. CoKE outperforms all the competitive baselines on FB15k and FB15k-237, and obtains comparable results as the best of them on the other two datasets. CoKE is also the most stable among the methods. It performs consistently the best (or near the best) on all the datasets, while the baselines fail to do so (e.g., pLogicNet* which performs quite well on FB15k underperforms on FB15k-237/WN18RR; TuckER and RotatE which perform near the best on these two datasets obtain substantially worse results on FB15k). The results demonstrate the effectiveness of CoKE in single-hop reasoning.

Parameter Efficiency We investigate parameter efficiency of CoKE. For comparison, we consider RotatE (Sun et al., 2019) and TuckER (Balažević et al., 2019b), which achieve previous state-of-the-art results with their optimal configurations explicitly stated. Table 4 presents the results on the four benchmarks. For each method, we report the number of parameters associated with the optimal configuration that leads to the performance shown in Tables 2 and 3.

Though a Transformer structure is used, CoKE is still parameter efficient, achieving better results with fewer parameters on FB15k/FB15k-237, and comparable results with a relatively small number of parameters on WN18/WN18RR. The reason is that compared with the rather small Transformer structure (6 layers with 4 attention heads), entity embeddings contribute most to the parameters due to a large vocabulary size. As entity embeddings are required by all the methods, their size becomes key to parameter efficiency. CoKE is able to work well with a small embedding size $D = 256$ on all the datasets.

4.2 Path Query Answering

This task is to answer path queries on KGs (Guu et al., 2015). A path query $s \rightarrow r_1 \rightarrow \dots \rightarrow r_k \rightarrow ?$ consists of an initial entity s and a sequence of relations r_1, \dots, r_k . Answering this question is to predict an entity o that can be reached from s by traversing r_1, \dots, r_k in turn. It is also formulated the same as our training task, i.e., entity prediction within paths. It degenerates to link prediction $s \rightarrow r_1 \rightarrow ?$ when $k = 1$.

Datasets We adopt the two datasets released by Guu et al. (2015), created from WordNet and Free-

	FB15k				WN18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
<i>Methods that use triples alone</i>								
Simple (Kazemi and Poole, 2018)	.727	.660	.773	.838	.942	.939	.944	.947
TorusE (Ebisu and Ichise, 2018)	.733	.674	.771	.832	.947	.943	.950	.954
ConvE (Dettmers et al., 2018)	.745	.670	.801	.873	.942	.935	.947	.955
ConvR (Jiang et al., 2019)	.782	.720	.826	.887	.951	.947	.955	.958
RotatE (Sun et al., 2019)	.797	.746	.830	.884	.949	.944	.952	.959
HypER (Balažević et al., 2019a)	.790	.734	.829	.885	.951	.947	.955	.958
TuckER (Balažević et al., 2019b)	.795	.741	.833	.892	.953	.949	.955	.958
<i>Methods that use graph contexts or rules</i>								
R-GCN+ (Schlichtkrull et al., 2017)	.696	.601	.760	.842	.819	.697	.929	.964
KBLRN (Garcia-Duran and Niepert, 2017)	.794	.748	—	.875	—	—	—	—
ComplEx-NNE+AER (Ding et al., 2018)	.803	.761	.831	.874	.943	.940	.945	.948
pLogicNet* (Qu and Tang, 2019)	.844	.812	.862	.902	.945	.939	.947	.958
CoKE (with triples alone)	.852	.823	.868	.904	.951	.947	.954	.960

Table 2: Link prediction results on FB15k and WN18. Baseline results are taken from original papers.

	FB15k-237				WN18RR			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
<i>Methods that use triples alone</i>								
ConvE (Dettmers et al., 2018)	.316	.239	.350	.491	.46	.39	.43	.48
ConvR (Jiang et al., 2019)	.350	.261	.385	.528	.475	.443	.489	.537
RotatE (Sun et al., 2019)	.338	.241	.375	.533	.476	.428	.492	.571
HypER (Balažević et al., 2019a)	.341	.252	.376	.520	.465	.436	.477	.522
TuckER (Balažević et al., 2019b)	.358	.266	.394	.544	.470	.443	.482	.526
<i>Methods that use graph contexts or rules</i>								
R-GCN+ (Schlichtkrull et al., 2017)	.249	.151	.264	.417	—	—	—	—
KBLRN (Garcia-Duran and Niepert, 2017)	.309	.219	—	.493	—	—	—	—
pLogicNet* (Qu and Tang, 2019)	.332	.237	.367	.524	.441	.398	.446	.537
CoKE (with triples alone)	.361	.269	.398	.547	.475	.437	.490	.552

Table 3: Link prediction results on FB15k-237 and WN18RR. Baseline results are taken from original papers.

	FB15k			FB15k-237		
RotatE	31.25M	.797	.884	29.32M	.338	.533
TuckER	11.26M	.795	.892	10.96M	.358	.544
CoKE	7.42M	.852	.904	7.03M	.361	.547
	WN18			WN18RR		
RotatE	40.95M	.949	.959	40.95M	.476	.571
TuckER	9.39M	.953	.958	9.39M	.470	.526
CoKE	13.76M	.952	.960	13.76M	.475	.552

Table 4: Parameter efficiency on the four benchmarks. Each cell reports number of parameters, MRR, H@10.

	WordNet	Freebase
Entities	38,551	75,043
Relations	11	13
Train Triples	110,361	316,232
Dev Triples	2,602	5,908
Test Triples	10,462	23,733
Train Paths	2,129,539	6,266,058
Dev Paths	11,277	27,163
Test Paths	46,577	109,557

Table 5: Number of entities, relations, triples and paths in each split of the two datasets.

base.² Triples of these two datasets are split into training and test sets, and paths have already been generated by random walks. Paths used for training are sampled from the graph composed of training triples alone, with the following procedure: (1) Uniformly sample a path length $k \in \{2, \dots, 5\}$ and a starting entity $s \in \mathcal{E}$. (2) Perform a random walk starting at s , continuing k steps by traversing r_1, \dots, r_k , and reaching entity o . (3) Output a path $s \rightarrow r_1 \rightarrow \dots \rightarrow r_k \rightarrow o$. Paths of length 1 are not sampled, but constructed by directly adding train-

ing triples. Paths used for test are generated from the whole graph containing both training and test triples, with the same procedure. Test paths which also appear as training instances are removed. See (Guu et al., 2015) for a detailed description of the dataset construction. Table 5 summarizes statistics of the two datasets.³

Training Details In this task, we train our model with *paths* from the *training* set (triples are paths

²<https://www.codalab.org/worksheets/0xfca41fdeec45f3bc6dddf31107b829f>

³The statistics reported here are calculated directly from the released data, which are slightly different from the numbers reported in the original literature (Guu et al., 2015).

of length 1). The maximum input sequence length is hence restricted to $K = 7$ (at most 5 relations between 2 entities). We use the same configuration for CoKE as in the link prediction task. We train with a learning rate of $\eta = 3e^{-4}$ and a batch size of 2048 for at most 20 epochs. We compute MQ (detailed later) over the dev set every 5 epochs, and select the epoch that leads to the best MQ.

Evaluation Protocol We follow the same evaluation protocol of (Guu et al., 2015), to make our results directly comparable. Specifically, for each test path $s \rightarrow r_1 \rightarrow \dots \rightarrow r_k \rightarrow o$ and the query $s \rightarrow r_1 \rightarrow \dots \rightarrow r_k \rightarrow ?$, we define: (1) candidate answers \mathcal{C} that “type match”, namely entities that participate in the final relation r_k at least once, i.e., $\mathcal{C} \triangleq \{o | \exists s' \text{ s.t. } (s', r_k, o) \in \mathcal{G}\}$; (2) correct answers \mathcal{P} that can be reached from s by traversing r_1, \dots, r_k , i.e., $\mathcal{P} \triangleq \{o | \exists e_1, \dots, e_{k-1} \text{ s.t. } (s, r_1, e_1), \dots, (e_{k-1}, r_k, o) \in \mathcal{G}\}$; (3) incorrect answers $\mathcal{N} \triangleq \mathcal{C} \setminus \mathcal{P}$. Here \mathcal{G} is the whole graph composed of training and test triples. Then we replace entity o with [MASK], feed the sequence into CoKE, and get the predicted distribution of o over all entities. We rank correct answer o along with incorrect answers \mathcal{N} according to the distribution probabilities in descending order, and compute the quantile as fraction of incorrect answers ranked after o . The quantile ranges from 0 to 1, with 1 being optimal. We report the mean quantile (MQ) aggregated over all test paths, and also the percentage of test cases with the correct answer o ranked in the top 10 (H@10).⁴

Main Results Table 6 reports the results of path query answering on the two datasets. As baselines, we choose compositional Bilinear, DistMult, and TransE devised by Guu et al. (2015), which model multi-hop paths by combining relations with additions and multiplications. We also compare with an improved Path-RNN (Das et al., 2017) and ROP (Yin et al., 2018), which combine relations with recurrent neural networks. We test our approach in five settings: CoKE (PATHS $\leq k$) for $k = 1, \dots, 5$, which means training with paths of length 1 to k . The $k = 5$ setting enables a fair comparison with

	WordNet		Freebase	
	MQ	H@10	MQ	H@10
Bilinear-COMP [†]	0.894	0.543*	0.835	0.421
DistMult-COMP [†]	0.904	0.311	0.848	0.386
TransE-COMP [†]	0.933	0.435	0.880	0.505
Path-RNN [‡]	0.989	—	—	—
ROP [#]	—	—	0.907	0.567*
CoKE (PATHS ≤ 1)	0.731	0.157	0.730	0.367
CoKE (PATHS ≤ 2)	0.914	0.490	0.889	0.570
CoKE (PATHS ≤ 3)	0.928	0.594	0.920	0.656
CoKE (PATHS ≤ 4)	0.939	0.643	0.935	0.719
CoKE (PATHS ≤ 5)	0.942	0.674	0.948	0.764

Table 6: Path query answering results on WordNet and Freebase. [†] Results reported from (Guu et al., 2015), [‡] results from (Das et al., 2017), and [#] results from (Yin et al., 2018).

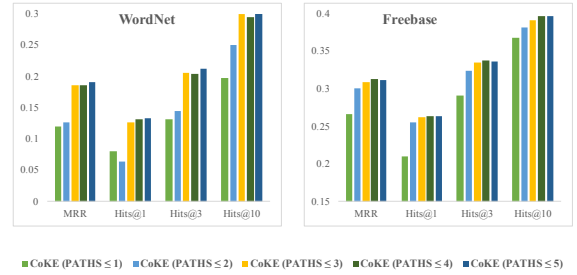


Figure 3: Link prediction results on length-1 test paths from WordNet (left) and Freebase (right).

the baselines, while the $k < 5$ settings actually use shorter paths for training.

As we can see, CoKE performs extremely well on this task. CoKE (PATHS ≤ 5) outperforms all the baselines (except for the MQ metric on WordNet), offering an absolute improvement in H@10 of up to 13.1% on WordNet and that of up to 19.7% on Freebase, compared against the current best-so-far (denoted by *). Notably, CoKE can achieve good results even if trained on relatively short paths. The $k = 3$ setting on WordNet and $k = 2$ setting on Freebase already outperform the baselines (in H@10) trained on longer paths of length up to 5. And the performance of CoKE grows significantly as the maximum path length k increases. The results demonstrate the superior capability of CoKE to model compositional patterns within paths so as to support multi-hop reasoning.

Further Analysis We further verify that training on multi-hop paths improves not only multi-hop reasoning but also single-hop reasoning. To do so, we consider a link prediction task on the two path query datasets. Specifically, we keep the training set unchanged (training paths of length 1 to 5), but

⁴The H@10 metric used here is slightly different from the one used in the link prediction task. Here incorrect answers are restricted to be entities that “type match”. But there is no such restriction in link prediction. We follow this definition of H@10 to make our results directly comparable to Guu et al. (2015). We refer readers to their evaluation script for details: <https://www.codalab.org/worksheets/0xf3ace41fdeec45f3bc6dddf31107b829f>.

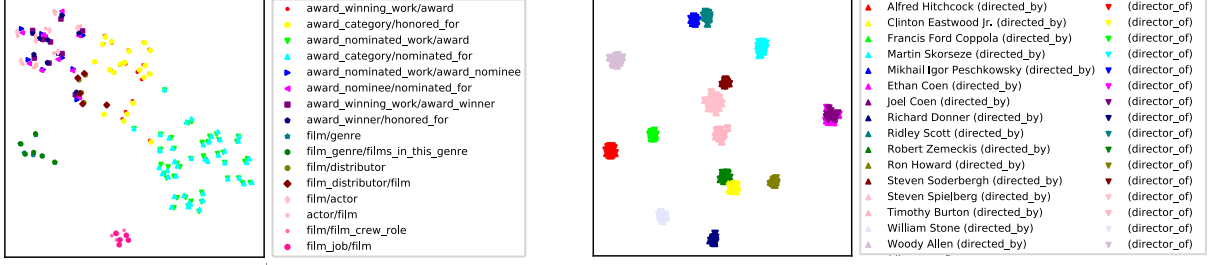


Figure 4: Contextualized representations of `TheKingsSpeech` (left) and `DirectorOf/DirectedBy` (right) learned by CoKE from FB15k. Each point is an entity/relation embedding within a triple. Different colors are used to distinguish different relations (left) or subjects/objects (right).

consider only test paths of length 1. For each test triple (s, r, o) , we create two prediction cases: $? \rightarrow r \rightarrow o$ and $s \rightarrow r \rightarrow ?$, and report aggregated MRR and $H@n$ for $n = 1, 3, 10$ (see § 4.1 for details).

We evaluate CoKE ($\text{PATHS} \leq k$) for $k = 1, \dots, 5$, which means training with paths of length up to k but test only on paths of length 1. The results are presented in Figure 3. We can see that the $k \geq 2$ settings significantly outperform the $k = 1$ setting in almost all metrics on both datasets (except for $k = 2$ in $H@1$ on WordNet). And the performance generally grows as k increases. The results verify that training on multi-hop paths further improves single-hop reasoning.

4.3 Visual Illustrations

This section provides visual illustrations of CoKE representations to show how they can distinguish contextual usage of entities and relations.

We choose entity `TheKingsSpeech` from FB15k as an example, collecting all triples where it appears. We feed these triples into the optimal CoKE model learned during link prediction, and get final hidden states of this entity, i.e., its contextualized representations within different triples. We visualize these representations in a 2D plot via t-SNE (Van der Maaten and Hinton, 2008), and show the result in Figure 4 (left). Here, a different color is used for each relation, and relations appearing less than 5 times are discarded. We can see that representations of this entity vary across triples, falling into clusters according to the relations. Similar relations, e.g., `award_winning_work/award_winner` and `award_nominated_work/award_nominee`, tend to have overlapping clusters. This indicates the capability of CoKE to distinguish fine-grained contextual meanings of entities, i.e., how the meaning of an entity varies across relations. Moreover, we observe that the two representations, one obtained

when this entity appears as a subject in (s, r, o) and the other as an object in (o, r^{-1}, s) , nearly coincide with each other in almost every case, where r^{-1} is the inverse relation of r , e.g., `film/genre` and `genre/films_in_this_genre`. This indicates that CoKE is pretty good at identifying relations and their inverse relations.

Figure 4 (right) further visualizes the contextualized representations of relation `DirectorOf` and its inverse relation `DirectedBy`, obtained in a similar way as the above case. Here, different colors are used to distinguish different directors. Directors appearing less than 10 times are discarded. Again, we observe that the two representations, one for `DirectorOf` in (s, r, o) and the other for `DirectedBy` in (o, r^{-1}, s) , nearly coincide in almost every case. And these representations fall into clusters according to directors. The two overlapping clusters (rightmost ones) correspond to `JoelCoen` and `EthanCoen`, referred to as the Coen brothers who write, direct and produce films jointly. This indicates the capability of CoKE to distinguish fine-grained contextual meanings of relations, i.e., how the meaning of a relation varies across entities.

5 Conclusion

This paper introduces *Contextualized Knowledge Graph Embedding* (CoKE), a novel paradigm that learns dynamic, flexible, and fully contextualized KG embeddings. Given an edge or a path formalized as a sequence of entities and relations, CoKE employs Transformer encoder to obtain contextualized representations for its components, which are naturally adaptive to the input, capturing contextual meanings of entities and relations therein. CoKE is conceptually simple yet empirically powerful, achieving new state of the art results in link prediction and path query answering on a number of widely used benchmarks. Visualization further

demonstrates that CoKE representations can indeed discern fine-grained contextual meanings of entities and relations.

As future work, we would like to (1) Investigate the effectiveness of different path definitions, e.g., those with intermediate entities. (2) Generalize CoKE to other types of graph contexts beyond edges and paths, e.g., subgraphs in arbitrary forms. (3) Apply CoKE to more downstream tasks, not only those within a given KG, but also those scaling to broader domains like computer vision and natural language understanding.

References

- Ivana Balažević, Carl Allen, and Timothy M. Hospedales. 2019a. Hypernetwork knowledge graph embeddings. In *ICANN*, pages 553–565.
- Ivana Balažević, Carl Allen, and Timothy M. Hospedales. 2019b. TuckER: Tensor factorization for knowledge graph completion. *arXiv:1901.09590*.
- Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *EMNLP*, pages 615–620.
- Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.
- Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. 2017. Global RDF vector space embeddings. In *ISWC*, pages 190–207.
- Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *EACL*, pages 132–141.
- Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *AAAI*, pages 1811–1818.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.
- Boyang Ding, Quan Wang, Bin Wang, and Li Guo. 2018. Improving knowledge graph embedding using simple constraints. In *ACL*, pages 110–121.
- Takuma Ebisu and Ryutaro Ichise. 2018. TorusE: Knowledge graph embedding on a lie group. In *AAAI*, pages 1819–1826.
- Jun Feng, Minlie Huang, Yang Yang, and xiaoyan zhu. 2016. GAKE: Graph aware knowledge embedding. In *COLING*, pages 641–651.
- Alberto Garcia-Duran and Mathias Niepert. 2017. KBLRN: End-to-end learning of knowledge base representations with latent, relational, and numerical features. *arXiv:1709.04676*.
- Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*, pages 855–864.
- Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *EMNLP*, pages 318–327.
- Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL-IJCNLP*, pages 687–696.
- Xiaotian Jiang, Quan Wang, and Bin Wang. 2019. Adaptive convolution for multi-relational learning. In *NAACL-HLT*, pages 978–987.
- Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *NIPS*, pages 4284–4295.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.
- Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *MACH LEARN*, 81(1):53–67.
- Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*, pages 705–714.
- Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *J MACH LEARN RES*, 9(85):2579–2605.
- Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*, pages 6294–6305.
- Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL-HLT*, pages 327–333.
- Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A review of relational machine learning for knowledge graphs. *PROC IEEE*, 104(1):11–33.
- Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016b. Holographic embeddings of knowledge graphs. In *AAAI*, pages 1955–1961.

- Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *SIGKDD*, pages 701–710.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237.
- Meng Qu and Jian Tang. 2019. Probabilistic logic neural networks for reasoning. *arXiv:1906.08495*.
- Sebastian Riedel, Limin Yao, Andrew Mccallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT*, pages 74–84.
- Petar Ristoski and Heiko Paulheim. 2016. RDF2Vec: RDF graph embeddings for data mining. In *ISWC*, pages 498–514.
- Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *arXiv:1703.06103*.
- Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge graph embedding by relational rotation in complex space. *arXiv:1902.10197*.
- Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *ACL Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66.
- Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.
- 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. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119.
- Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *EMNLP*, pages 1366–1371.
- Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. TransG: A generative model for knowledge graph embedding. In *ACL*, pages 2316–2325.
- An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. 2019. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *ACL*, pages 2346–2357.
- 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 *ICLR*.
- Wenpeng Yin, Yadollah Yaghoobzadeh, and Hinrich Schütze. 2018. Recurrent one-hop predictions for reasoning over knowledge graphs. In *COLING*, pages 2369–2378.