Triple Classification Using Regions and Fine-Grained Entity Typing

Tiansi Dong
B-IT, University of Bonn, Germany
dongt@bit.uni-bonn.de

Zhigang Wang
Aisino Corporation, China
wangzhigang@aisino.com

Juanzi Li
DCST, Tsinghua University, China
lijuanzi@tsinghua.edu.cn

Christian Bauckhage,[1,2] **Armin B. Cremers**[1]
[1]B-IT, University of Bonn, Germany
[2]Fraunhofer IAIS, Germany
{bauckhag, abc}@bit.uni-bonn.de

Abstract

A Triple in knowledge-graph takes a form that consists of *head, relation, tail*. Triple Classification is used to determine the truth value of an unknown Triple. This is a hard task for 1-to-N relations using the vector-based embedding approach. We propose a new region-based embedding approach using fine-grained type chains. A novel geometric process is presented to extend the vectors of pre-trained entities into n-balls (n-dimensional balls) under the condition that *head* balls shall contain their *tail* balls. Our algorithm achieves zero energy cost, therefore, serves as a case study of perfectly imposing tree structures into vector space. An unknown Triple (h, r, x) will be predicted as true, when x's n-ball is located in the r-subspace of h's n-ball, following the same construction of known *tails* of *h*. The experiments are based on large datasets derived from the benchmark datasets WN11, FB13, and WN18. Our results show that the performance of the new method is related to the length of the type chain and the quality of pre-trained entity-embeddings, and that performances of long chains with well-trained entity-embeddings outperform other methods in the literature. Source codes and datasets are located at https://github.com/GnodIsNait/mushroom.

Introduction

Knowledge-graphs represent truth knowledge in the Triple *(head, relation, tail)*, shortened as (h, r, t). In Semantic Web Community, such a Triple is named as *(subject, predicate, object)*. Knowledge-graphs such as Word-Net, Yago, and Freebase (Miller 1995; Suchanek, Kasneci, and Weikum 2007; Bollacker et al. 2008) are very useful for AI applications, e.g. question-answering, query expansion, information retrieval, document classification (Manning, Raghavan, and Schütze 2008; Socher et al. 2013; Bordes et al. 2013; Wang et al. 2014a; Wang and Li 2016). However, knowledge-graphs normally suffer from incompleteness. One research topic in AI is to predict the missing part of a knowledge-graph. The basic task is Triple Classification, which is to determine the truth value, or the degree of truth value, of an unknown Triple.

In the literature of representational learning, a Triple (h, r, t) is encoded as a quasi triangular relation $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, which can be understood as a *translation* from $\mathbf{h}$ to $\mathbf{t}$ by adding $\mathbf{r}$. Using this method, (Bordes et al. 2013) achieved the state-of-the-art performance, but it suffers from the 1-to-N, N-to-1 and N-to-N relations. Hyperplane projection methods allow an entity to have several embeddings for different relations. However, these methods do not significantly improve the performance of predicting unknown tails of Triples in 1-to-N relations (Wang et al. 2014b; Lin et al. 2015).

It is not difficult to understand that there is no perfect vector representation for 1-to-N relation: Let $\mathbf{isa}$ be the perfect vector representation for the *isa* relation, $(zurich, isa, city)$ and $(new\_york, isa, city)$ be two Triples. Ideally, we shall have $\mathbf{zurich} + \mathbf{isa} = \mathbf{city}$ and $\mathbf{new\_york} + \mathbf{isa} = \mathbf{city}$. However, this leads to a wrong assertion that $zurich$ and new_york are the same city. We propose to extend entity-embeddings from vectors into regions, so that the *isa* relation can be implicitly represented by the inclusion relation between regions: *zurich* region and *new_york* region are located inside the *city* region, the *city* region is inside the *municipality* region, ... We can have a further fine-grained typing chain of *city*: *city*, *municipality*, *region*, *physical_entity*, *entity*, as illustrated in Figure 1. Such chains can be interpreted as an *incremental conceptual clustering* (Fisher 1987), and proven to be useful. They carry more type information than a single type (Ren et al. 2016), provide more specific semantic information (Xu et al. 2016). They have benefitted many real applications, e.g., knowledge-base completion (Dong et al. 2014), entity linking (Ling, Singh, and Weld 2015; Durrett and Klein 2015), relation extraction (Liu et al. 2014), and question-answering (Yahya et al. 2014).

In this article, we propose a new region-based entity-embedding method using type chains. Given pre-trained vector entity-embeddings, we extend them into regions, in which *tail* regions of 1-to-N relations are located in the *head* region. A type chain will introduce nested regions. As an entity can have different 1-to-N relations, we need to distinguish different kinds of inclusion relations among regions. For example, given two Triples *(beijing, isa, city)* and *(inner_city, part_of, city)*. Both the *beijing* ball and the *inner_city* ball shall be located inside the *city* ball. We need to distinguish the *isa* relation from the *part_of* relation. Our solution is to introduce the *isa* subspace and the *part_of* subspace into the *city* region. The *beijing* ball is located inside the *isa* subspace, while *inner_city* ball is inside the *part_of* subspace. The two subspaces are disconnected from each other, as illustrated in Figure 2. This method can be understood as a recursive usage of supporting vector machines (SVM) with n-balls as kernels, i.e. (Shawe-Taylor and Cristianini 2004).

We use vectors trained by TEKE model (Wang and Li 2016) as pre-trained entity-embeddings, and extend them into balls in n-dimensional spheres, namely n-balls. Given tails $t_1, t_2, \ldots, t_n$ of head h with relation r, and fine grained typing chain of $h_0(=h), h_1, h_2, \ldots, h_k$, we design a new geometrical process to construct n-balls of t_is and h_js, so that any two n-balls of t_i are disconnected, t_i's n-ball is inside the r-subspace of h_0's n-ball, and n-ball of h_{i-1} is contained by the *isa* subspace of h_i's n-ball. The relations of containment shall be strictly realised. To determine whether a new entity x is a tail of head h_0 with relation r, we apply the same geometric process to x. If the n-ball of x is contained by the r-subspace of n-ball of h, (h, r, x) will be predicted as positive.

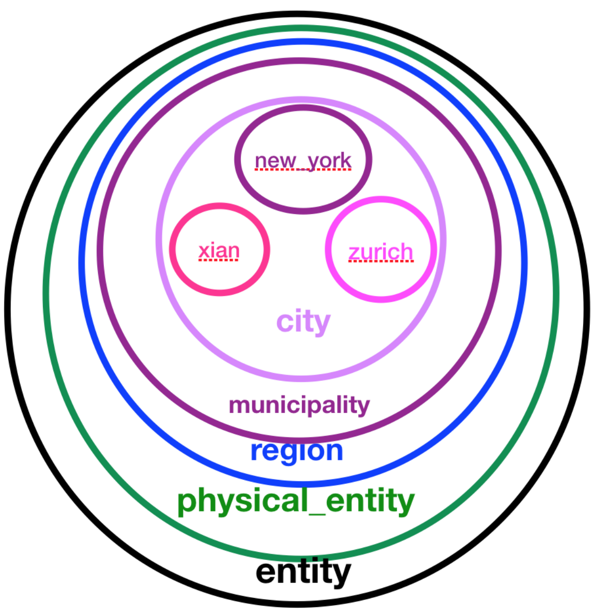

Figure 1: n-ball embedding

The contributions of this article are as follow: (1) a novel graph-embedding method is proposed to use n-ball for entity-embedding, and implicitly represent 1-to-N relations as inclusion relations among n-balls. Instead of a single type, we use fine-grained entity typing in Triple Classification; (2) a geometric transformation is proposed to strictly encode all selected Triple relations into n-ball embeddings, which cannot be achieved by the back-propagation method (Rumelhart, Hinton, and Williams 1988).

The rest of the article is structured as follows: Section 2 reviews related works; Section 3 presents our method; Section 4 shows experiment results; Section 5 summarizes the current work, lists on-going works, and the impact.

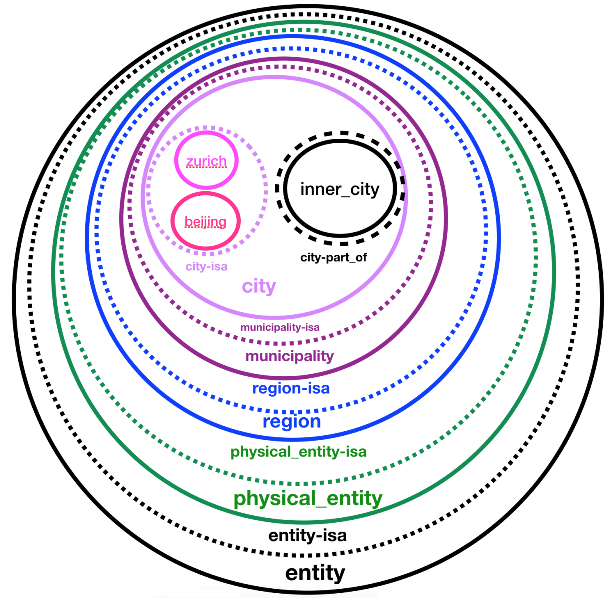

Figure 2: n-ball embedding with subspaces

Related Works

Triple Classification using representational learning is related to researches in the fields of graph embedding and reasoning with loss-function.

Graph Embedding

Representational learning has been widely applied for knowledge-graph representation and reasoning. TransE (Bordes et al. 2013) trained vector representations of (h, r, t), in which the vector triangle equation $\mathbf{t} \approx \mathbf{h} + \mathbf{r}$ stand. TransH (Wang et al. 2014b) relaxed this approximation relation on projection hyperplanes, i.e. $\mathbf{t}_\perp \approx \mathbf{d}_r + \mathbf{r}_\perp$. In TransR (Lin et al. 2015) and TransD (Ji et al. 2015), the vector triangle approximation is further updated to $M_r\mathbf{h} + \mathbf{r} \approx M_r\mathbf{t}$, that is, the approximation holds, after the head vector and the tail vector are transformed by a relation-related matrix M_r. In TransD (Ji et al. 2015) transformation matrix M_r is determined by both entities and relations. These models can be understood as the extension of TransE using the kernel approach (Shawe-Taylor and Cristianini 2004). CTransR (Lin et al. 2015) and TransG (Xiao et al. 2015b) extend TransR by clustering $\mathbf{h}$ and $\mathbf{t}$ before applying matrix transformation. The vector triangular approximation holds for representative vectors within its cluster in the kernel space, i.e. $M_r\mathbf{h}' + \mathbf{r} \approx M_r\mathbf{t}'$, where $\mathbf{h}'$ and $\mathbf{t}'$ are the representative vectors of $\mathbf{h}$ and $\mathbf{t}$ in their clusters, respectively. TransA (Xiao et al. 2015a) introduced a weight matrix W_r: $f_r(h, t) = (|\mathbf{h} + \mathbf{r} - \mathbf{t}|^\top) W_r (|\mathbf{h} + \mathbf{r} - \mathbf{t}|)$.

Representation learning only using Triples from knowledge-base cannot predict entities which do not exist in the knowledge-base. This limitation can be approached by introducing text information into the training process, e.g. (Socher et al. 2013; Wang et al. 2014a; Zhong et al. 2015; Zhang et al. 2015;

Xie et al. 2016). For example, by combining a CBOW and TransE to learn Triple embeddings, or by adopting deep convolution neural model to maximize the prediction of entity descriptions, experiment results are significantly improved, e.g., (Wang and Li 2016; Han, Liu, and Sun 2016).

To improve the performance of the reasoning with 1-to-N relations, researchers have proposed region-based graph embedding methods. The KG2E (He et al. 2015) model embeds entities using Gaussian distributions, namely high-dimensional regions of probability. (Xiao, Huang, and Zhu 2016) uses manifolds: a Triple (h, r, t) is interpreted as a manifold $\mathcal{M}(\mathbf{h}, \mathbf{r}, \mathbf{t}) = D_r^2$, that is, given h and r, tail t shall be located in the manifold. (Nickel and Kiela 2017) uses Poincaré balls to represent entities and relations, as an unsupervised approach to embed tree structures.

Loss-Function Using Margin and Negative Samples

The dominant method for representation training is to minimize a loss-function (LeCun et al. 2006), which tries to guarantee that the triangular relation of true Triples is better than that of negative samples within a margin (Gutmann and Hyvärinen 2012). Formally, let δ be the margin, $\mathcal{S}$ be the set of true Triples, and $\mathcal{S}'$ be the set of negative samples, the loss-function is written as follows:

$$L = \sum_{(h,r,t)\in\mathcal{S}} \sum_{(h,r,t')\in\mathcal{S}'} \max(0, f(h,r,t) + \delta - f(h,r,t'))$$

where $f(x, y, z) \triangleq \|\mathbf{x} + \mathbf{y} - \mathbf{z}\|$ is a degree measurement of the triangular relation among vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$. Stochastic Gradient Descent (SGD) process is applied to reduce the value of L untill a local minimum is reached.

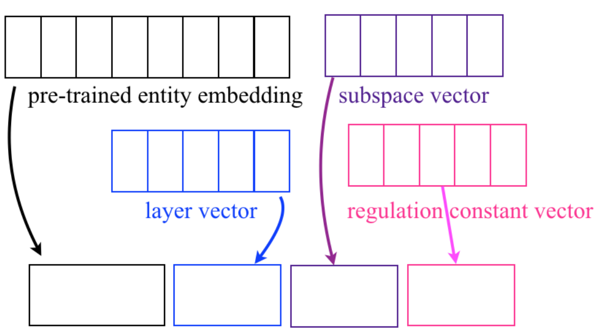

Figure 3: Four components of the central point

Triple Classification Using n-Ball Embeddings

Loss-function plus SGD does not guarantee the global minimum. Negative sampling shall not exhaust all false Triples. We distinguish from 'samplings' the 'type', which can be intuitively understood as a bounded region that separates all negative samples from true Triples. To simplify the process, we restrict these regions to n-balls, and develop geometric construction approach to achieve the global minimum.

We start with some terminologies. The n-ball embedding of entity e is written as $\mathbb{B}(\mathbf{O}_e, rd_e)$, where $\mathbf{O}_e$ is the vector of the central point of the n-ball of entity e, and rd_e is the radius of this n-ball. $\mathbf{O}_e$ can be further specified by $(\mathbf{d}_e, l_e)$, where $\mathbf{d}_e$ is the unit vector representing the direction of $\mathbf{O}_e$, and l_e is the length of $\mathbf{O}_e$. We define n-ball as an open space as follows: $\mathbb{B}(\mathbf{O}_e, rd_e) \triangleq \{\mathbf{p} | \|\mathbf{O}_e - \mathbf{p}\| < rd_e\}$, in which $\|\mathbf{x} - \mathbf{y}\| \triangleq \sqrt{(\mathbf{x} - \mathbf{y})^\top(\mathbf{x} - \mathbf{y})}$. $\mathbb{B}(\mathbf{O}_e, rd_e)$ is the set of vector $\mathbf{p}$ whose Euclidean distance to $\mathbf{O}_e$ is less than rd_e. That is, n-balls are open spaces. The complement region of an n-ball is not introduced (Dong 2008).

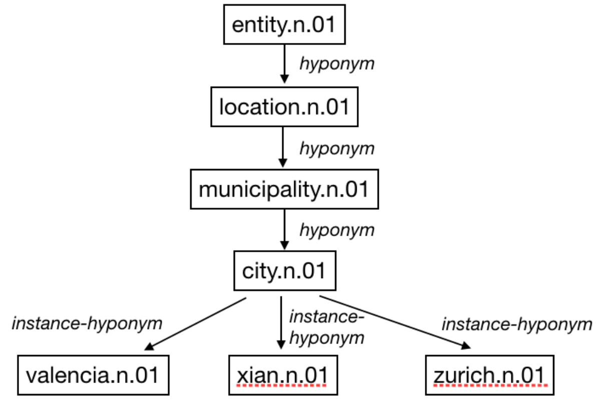

Figure 4: A type chain for three concrete cities

The Structure of the Central Point

The structure of the central point consists of four pieces of vector information as illustrated in Figure 3.

- the vector embedding of e acquired from corpus;
- the length p of the type chain of e is encoded by a layer vector, whose first p elements are '1', followed by '0'. For example, the layer vector of entity.n.01 is [0,0,0,0,0], the layer vector of location.n.01 is [1,0,0,0,0], the layer vector of xian.n.01 is [1,1,1,1,0], see Figure 4. Siblings have the same layer vector;
- different relations to e's parent are encoded by different vectors, which are called *subspace vectors*.
- regulation constant vector, whose function is to avoid an n-ball containing the origin point of the space.

Inspired by (Zeng et al. 2014) and (Han, Liu, and Sun 2016), we concatenate the pre-trained entity-embedding vector, the subspace vector, the layer vector, and the regulation constant vector as the central point of an n-ball.

Construct n-Balls and Subspaces

Given a Triple (h, r, t), we construct the r-subspace within $\mathbb{B}(\mathbf{O}_h, rd_h)$, which is also an n-ball, written as $\mathbb{B}(\mathbf{O}_{h_r}, rd_{h_r})$. For each tail t satisfying (h, r, t), $\mathbb{B}(\mathbf{O}_t, rd_t)$ is part of the r-subspace $\mathbb{B}(\mathbf{O}_{h_r}, rd_{h_r})$. We define two predicates for *being_disconnected* ($\mathbf{DC}$) and *being_inside* ($\mathbf{P}$) relations as follows:

- $\mathbb{B}(\mathbf{O}_{e_1}, rd_{e_1})$ disconnects from $\mathbb{B}(\mathbf{O}_{e_2}, rd_{e_2})$, written as $\mathbf{DC}(e_1, e_2)$, if the distance between $\mathbf{O}_{e_1}$ and $\mathbf{O}_{e_2}$ is equal

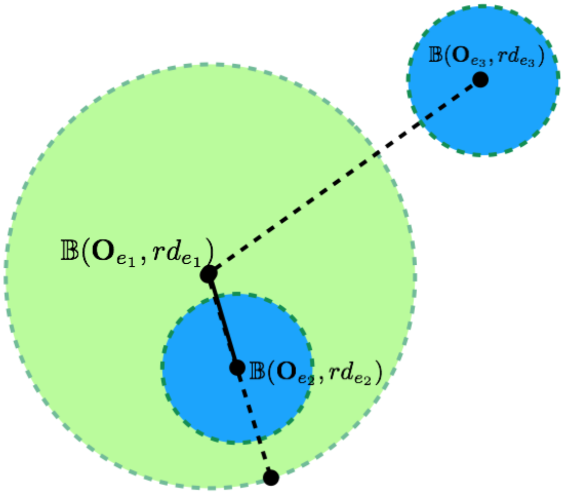

Figure 5: $\mathbb{B}(\mathbf{O}_{e_2}, rd_{e_2})$ is inside $\mathbb{B}(\mathbf{O}_{e_1}, rd_{e_1})$, $\mathbb{B}(\mathbf{O}_{e_1}, rd_{e_1})$ disconnects from $\mathbb{B}(\mathbf{O}_{e_3}, rd_{e_3})$

to or greater than the sum of rd_{e_1} and rd_{e_2}.

$$\mathbf{DC}(e_1, e_2) \triangleq \|\mathbf{O}_{e_1} - \mathbf{O}_{e_2}\| \geq rd_{e_1} + rd_{e_2} \quad (1)$$

- $\mathbb{B}(\mathbf{O}_{e_1}, rd_{e_1})$ is inside $\mathbb{B}(\mathbf{O}_{e_2}, rd_{e_2})$, written as $\mathbf{P}(e_1, e_2)$, if rd_{e_2} equals to or is greater than the sum of rd_{e_1} and the distance between $\mathbf{O}_{e_1}$ and $\mathbf{O}_{e_2}$, as illustrated in Figure 5.

$$\mathbf{P}(e_1, e_2) \triangleq rd_{e_2} \geq \|\mathbf{O}_{e_1} - \mathbf{O}_{e_2}\| + rd_{e_1} \quad (2)$$

We need to strictly encode all known Triple relations into inclusion relations among n-balls. This is a big challenge, as the widely adopted back-propagation training process quickly reaches a non-zero local minimum and terminates[1] (Rumelhart, Hinton, and Williams 1988). The problem is that when locations and sizes of two n-balls are updated locally, already improved locations and sizes will very easily deteriorate. We, therefore, use the classic depth-first recursion process, and employ three transformations to construct and train n-ball embeddings as follows.

- Homothethic transformation, which keeps the direction of the central point, and enlarges lengths of the central vector l and the radius rd with the same rate k ($k > 0$).
- Shifting transformation, which keeps the length of the radius rd of an n-ball, and add a new vector $\mathbf{p}$ to its central point.
- Rotation transformation, which keeps the length of the radius rd, and rotates angle α with the i-th and the j-th elements of $\mathbf{O}$.

[1]This happens even for small datasets. A toy dataset was created to show that the target configuration could not be achieved by back-propagation process. The source code is available at https://github.com/GnodIsNait/bp94nball

Algorithm 1: construct n-ball embeddings

```
input  : known tails: (h, r, t_1)...(h, r, t_M), t_i ∈ tails;
         type chain: tChains =[h, h_1, ..., h_N];
         pre-trained entity-embeddings: EV
output: r-subspace of n-ball of head h: ballHr;
         geometric transformation history: tranHis

tranHis = []
// initialise n-balls of [t_1, ...,t_M]
foreach ele ∈ [t_1, ...,t_M] do
  | bTails[ele]=init_nball(ele, EV)
end
// make n-balls of [t_1, ...,t_M]
   mutually be disconnected using
   three geometric transformations
bTails, tranHis =disconnect(bTails, tranHis)
ballHr =init_nball(h, EV)
// for each member in bTails, create
   n-ball of r-space of h to contain,
   using geometric transformations
foreach ele ∈ bTails do
  | bHr [ele ], tranHis
  |  =contain(ballHr, r, ele, tranHis)
end
while tChains not empty do
  | ele = pop(tChains, 0)
  | if ele ==h then
  |   | // create the minimal cover of
  |   |    bHr as the isa-subspace
  |   |    inside h's n-ball
  |   | ballHr =mini_cover_ball(bHr)
  |   | bTs[ele_0] = ballHr
  | else
  |   | bTs [ele ]=init_nball(ele, EV)
  |   | bTs [ele ], tranHis
  |   |  =contain(bTs[ele], isa, bTs[ele_0], tranHis)
  |   | ele_0 = ele
  | end
  | return ballHr, tranHis
end
```

Homothethic transformation keeps the pre-trained word-embeddings, while shifting and rotation transformations change pre-trained word-embeddings. To keep the pre-trained vectors, we apply the three transformations with different priorities: homothethic transformation with the highest priority, followed by shifting transformation, then rotation transformation. To prevent already trained relations from deteriorating, we introduce the principle of family action: *if one transformation is applied for an n-ball, the same transformation shall be applied for all its child balls*. We use depth-first recursive procedure to construct h's n-ball and its r-subspaces, as illustrated in Algorithm 1. Subspaces of h are mutually disconnected. The n-ball of h is the minimal cover of all its r-subspaces.

Algorithm 2: $Triple\_predict(\mathsf{x}, \mathsf{h}, \mathsf{r}, \mathsf{KG}, \gamma, \mathsf{EV})$: whether the Triple $(\mathsf{h}, \mathsf{r}, \mathsf{x})$ holds in knowledge-graph KG

```
input : head h, relation r, entity x, knowledge-graph
        KG, ratio γ, entity-embeddings EV
output: True, if n-ball of x is contained by n-ball of h;
        False, otherwise
// get all known tails of h related
   with r in KG
tails = get_all_known_tails(h, r, KG);
if number_of(tails) > 0 then
    // get the fine grained type chain
       of h in KG
    tChains = get_fine_grained_type_chain(h, KG)
       // construct the model, return
       (1) n-ball of h, (2) the
       initialisation parameter, the
       sequence of geometric
       transformations applied for tails

    ballH,tranHis = nballs(h, r, tails, tChains, EV)
    ballH = enlarge_radius(ballH, γ)
    // construct n-ball of x with
       tranHis
    ballX = construct_nball(x, tranHis, EV)
    if contained_by(ballX, ballH) then
        return True;
    else
        return False;
    end
else
    return False;
end
```

Experiments and Evaluation

The Aim of the Experiments

The aim of the experiments is to evaluate the proposed method for classifying Triples of 1-to-N relations. Given a new Triple (h, r, x), we will construct a Triple-predicting model $\mathcal{M}(h, r)$ from the knowledge-graph, and predict whether (h, r, x) is true by inspecting whether the n-ball of x is inside the r-subspace of h's n-ball.

Datasets

Knowledge-graphs for Triple Classification are normally generated from WordNet (Miller 1995), and Freebase (Bollacker et al. 2008). WordNet is a large English lexical database, whose entities are called *synset* representing a distinct word sense. Freebase is a large knowledge graph about world facts. Following the evaluation strategies, e.g. (Bordes et al. 2013; Socher et al. 2013; Wang and Li 2016; Ji et al. 2015), we use WN11, WN18, FB13 to generate datasets for classifying Triples of 1-to-N relations. We have manually analyzed all the relations in the three datasets, and rename them in term of the containment relation of n-balls as follows.

id	WN18/WN11/FB13 relation	n-ball relation
0	A B _member_of_domain_topic A _domain_topic B	$contain_0$(A, B)
1	A B _member_meronym	$tr\_contain_1$(A, B)
2	A B _member_of_domain_region A _domain_region B	$contain_2$(A, B)
3	A B _hypernym A _type_of B	$tr\_contain_3$(B, A)
4	A B _member_holonym	$tr\_contain_1$(B, A)
5	A B _instance_hypernym A _subordinate_instance_of B	$contain_5$(B, A)
6	A B _member_of_domain_usage	$contain_6$(A, B)
7	A B _synset_domain_topic_of A _synset_domain_topic B	$contain_0$(B, A)
8	A B _hyponym	$tr\_contain_3$(A, B)
9	A B _instance_hyponym A _has_instance B	$contain_5$(A, B)
10	A B _synset_domain_usage_of	$contain_6$(B, A)
11	A B _has_part	$tr\_contain_{11}$(A, B)
12	A B _part_of	$tr\_contain_{11}$(B, A)
13	A B _synset_domain_region_of	$contain_2$(B, A)
14	A gender B	$contain_{14}$(B, A)
15	A nationality B	$contain_{15}$(B, A)
16	A profession B	$contain_{16}$(B, A)
17	A place_of_death B	$contain_{17}$(B, A)
18	A place_of_birth B	$contain_{18}$(B, A)
19	A location B	$contain_{19}$(B, A)
20	A institution B	$contain_{20}$(B, A)
21	A cause_of_death B	$contain_{21}$(B, A)
22	A religion B	$contain_{22}$(B, A)
23	A parents B	$contain_{23}$(B, A)
24	A children B	$contain_{24}$(A, B)
25	A ethnicity B	$contain_{25}$(B, A)

Table 1: mapping WN18/WN11/FB13 relations to n-ball relations

Dataset	$\#\mathcal{R}$	$\#\mathcal{E}$	#train Triple	#test Triple
FB13-$n$ball	13	75,043	306,747	45,897
WN11-$n$ball	5	38,696	94,472	14,587
WN18-$n$ball	7	40,943	60,490	13,148

Table 2: Datasets extracted from WN11, FB13, WN18

- Each 1-to-N relation is assigned by an identification number from 0 to 25. If r is the ith non-symmetric relation, Triple (h, r, t) is named as $contain_i(h, t)$ in n-ball representation, the prefix $tr\_$ is added to $contain_i$, if r is transitive;
- if r^{-1} is the inverse of r, Triple (t, r^{-1}, h) will be transformed to (h, r, t).

The whole relations are listed in Table 1. We transform Triples in datasets into n-ball representations, and remove duplicated Triples. WN11 dataset has 11 relations, 38,696 entities, 112,581 training Triples, 2,609 valid Triples, 21,088 testing Triples. In the literature of machine learning, valid dataset is used to adjust hyperparameters of models (Bishop 2006; Goodfellow, Bengio, and Courville 2016). Our Triple prediction model does not have hyperparamers, which need to be tuned by sample data, so we integrate true Triples in the valid dataset into the training dataset. After transformation and reduction, WN11-$n$ball dataset has 5 relations, totaling 94,472 training Triples (91,888 from WN11 training Triples, 2,584 from WN11 valid Triples), and 20,495 testing Triples; applying the same data-processing, WN18 dataset only remains 135 testing Triples. We increased the number of testing Triples to 6,574 with true values, and the same number of Triples with false values, following the same setting used for WN11 (Socher et al. 2013). FB13 dataset has 75,043 entities, 306,747 training Triples, and 45,897 testing Triples.

To predict truth-value of (h, r, x), our model must have at least one tail t of (h, r, t) in the training set. We remove all Triples in the testing set, which does not have sample tails in the training set. Final dataset sizes are listed in Table 2.

Experimental Results for Triple Classification

Implementation. For each testing Triple (h, r, t), we construct a Triple-predicting model $\mathcal{M}(h, r)$ with h's hypernym path and tails of h in the training set, and record the transformation sequence. Given a knowledge-graph $\mathcal{KG}$, we used entity-embeddings trained in two different ways: (1) only $\mathcal{KG}$, i.e. TransE (Bordes et al. 2013), and (2) $\mathcal{KG}$ with text corpus, i.e. TEKE (Wang and Li 2016).

Evaluation Protocol. In transformation-based approach, a threshold δ_r is defined to evaluate a score function f: if a transformation score f of a Triple is below δ_r, this Triple will be predicted as true. We apply this method for the n-ball setting as follows: To judge whether x is the tail of h with relation r: (h, r, x), we initialize the n-ball of x, $\mathbb{B}(\mathbf{O}_x, rd_x)$, the same way as initializing other n-balls, and transform $\mathbb{B}(\mathbf{O}_x, rd_x)$ following the same sequence of transformations as the n-ball construction process of any true tail t of h with relation r, (h, r, t). If the final $\mathbb{B}(\mathbf{O}_x, rd_x)$ is located in the r-subspace of the n-ball of h, $\mathbb{B}(\mathbf{O}_{h_r}, rd_{h_r})$, (h, r, x) will be predicted as true, otherwise false. We update the final radius rd_{h_r} with a ratio γ to maximize performances of predicting results, as described in Algorithm 2.

Experiments with FB13-$n$ball Triples in FB13 dataset do not have hypernym relations. As a result, the length of the fine-grained typing chain is zero. This degrades the model into a trivial case: Tripe (h, r, x) holds, if and only if x is inside the r-subspace of h's n-ball. We draw the minimum r-subspace containing all known true tails. Using TEKE_E vectors, the accuracy reaches a maximum of 78%, which is almost the same as 77.4% reported in (Wang and Li 2016). Though our datasets are not exactly the same as the benchmark datasets, this result is consistent with existing results in the literature. FB13-$n$ball datasets and pre-trained entity-embeddings are free for downloads[2].

Experiments with WN11-$n$ball Triples in WN11 dataset have hypernym relations. Heights of type chains vary from 1 to 12. Precisions, recalls, and accuracies are illustrated in Figure 6, Figure 7, and Figure 8. We expand the range of γ from 0.6 to 2.3, to see whether smaller γ can contribute to precision. Experiment results support that smaller γ greatly improves precisions, and severely weakens recalls. The maximum value of 73% in accuracy is reached by using TEKE_E vectors when $\gamma = 1.4$, a bit less than 75.9% reported by (Wang and Li 2016).

We analyze the contribution of lengths of type chains to precision. Choose $\gamma = 1.0$, Figure 9 shows that lengths of type chains can contribute greatly to precision by using TEKE_E or TEKE_H entity-embeddings: If lengths are higher than 2, precision will be greater than 90%, and has a strong tendency to increase to 100% using both TEKE entity-embeddings. Such performance is not clear by TransE entity-embeddings, which suffers from a sudden drop with the length of type chains at 10. After examining the datasets, we find that there are only 19 testing records with type-chain length of 10, and that none of them is correctly predicted using TransE embeddings.

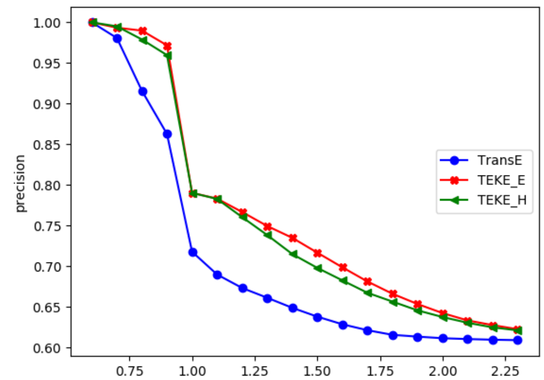

Figure 6: Precision of Triple prediction using WN11-$n$ball dataset. When γ increases, the precision will drop

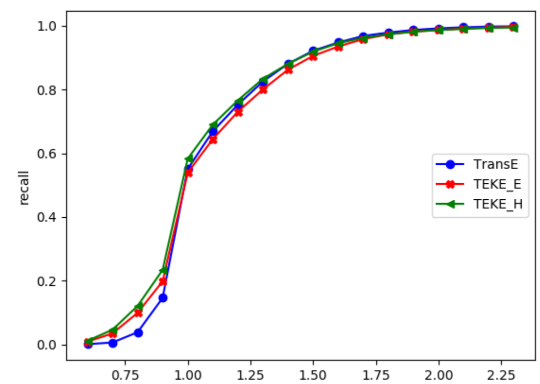

Figure 7: Recall of Triple prediction using WN11-$n$ball dataset. When γ increases, the recall will increase

[2] https://figshare.com/articles/FB13nball_zip/7294295

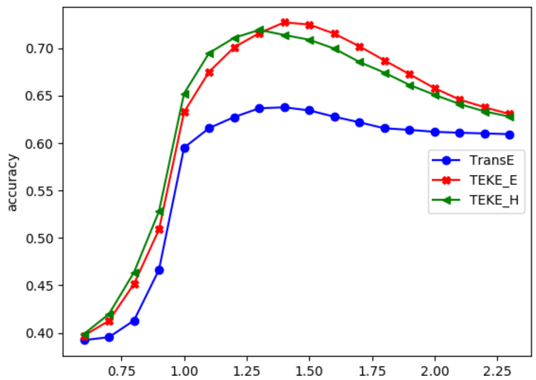

Figure 8: Accuracy of Triple classification using WN11-$n$ball with different γ values

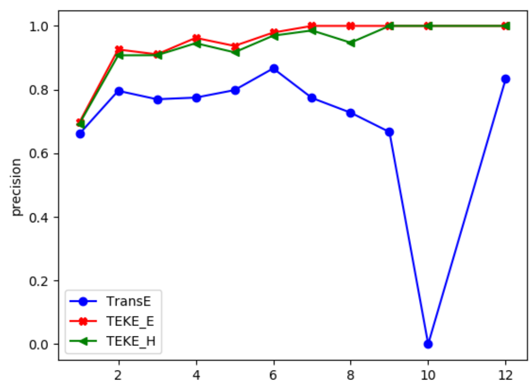

Figure 9: Precision vs. length of type chains in WN11-$n$ball Dataset, when $\gamma = 1$

We analyze the relation between accuracies and lengths of type chains. Figure 10 shows the maximum accuracy with regard to Triples whose lengths of type chains are higher than N. For example, if we choose $\gamma = 1.5$, the accuracy using TEKE_E embeddings will reach 81.8% for Triples whose type chains are longer than 8. When type chains are longer than 4, the accuracy of our model using TEKE_E embeddings will significantly outperform the results reported in (Wang and Li 2016).

The performance by using TransE is lower than and not as stable as that by using TEKE_E or TEKE_H entity-embeddings. The reason is that TEKE_E and TEKE_H entity-embeddings are jointly trained by knowledge-graph and corpus information, while TransE does not consider corpus information. WN11-$n$ball datasets and pre-trained entity-embeddings are free for public access[3].

Experiments with WN18-$n$ball Triples in WN18-$n$ball dataset have richer hypernym relations than Triples in

[3]https://figshare.com/articles/WN11nball/7294307

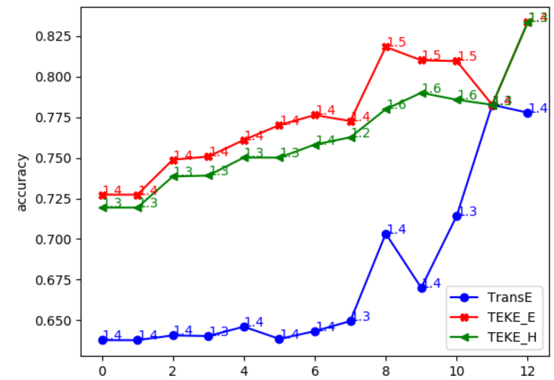

Figure 10: Accuracy vs. different lengths of type chains in WN11-$n$ball Dataset; when lengths increases, the accuracy has a tendency to increase; numbers in the plots represent the γ value with which the accuracy reaches maximum

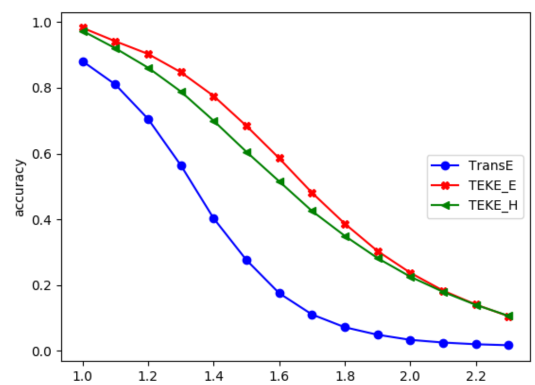

Figure 11: Using the Triple prediction model on the unballanced WN18-$n$ball testing dataset, the accuracy reaches 98% using TEKE_E pre-trained entity-embeddings, when $\gamma = 1$

WN11-$n$ball. For example, the lengths of type chains are longer than those of WN11-$n$ball: 57.1% type chains in WN18-$n$ball are longer than 5; the maximum length is 18. However, it turns out that only 72 tails of *true* testing Triples have pre-trained entity-embeddings, while 6574 tails of *false* testing Triples have. With this unballanced testing dataset, our predicting model produces surprisingly good accuracy. Accuracy reaches 98% using TEKE_E pre-trained entity-embeddings, when $\gamma = 1$, as illustrated in Figure 11. Lengths of type chains contribute to accuracy, see Figure 12. TransE embedding also delivered great results, though the performance remains less stable than that of TEKE_E and TEKE_H embeddings. WN18-$n$ball datasets and pre-trained entity-embeddings are free for downloads[4].

[4]https://figshare.com/articles/WN18nball/7294316

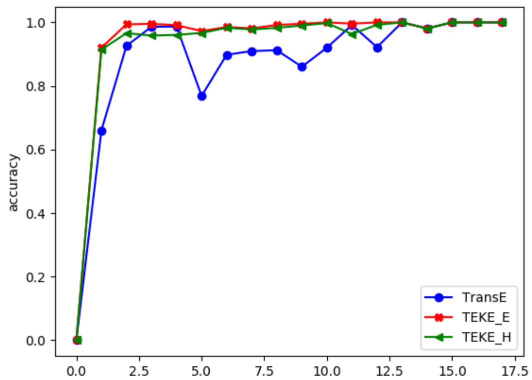

Figure 12: Using the Triple prediction model on the unballanced WN18-$n$ball testing dataset, the accuracy increases along with the length of type-chains, when $\gamma = 1$

Conclusions, Limitations, and Impact

Triple classification for 1-to-N relations is a tough problem. We propose a region-based embedding method using fine-grained entity typing. The contribution of our work is to perfectly encode 1-to-N symbolic relations into the embedding space, resulting in n-ball embeddings. Experiment results are surprisingly good for well pre-trained entities having long type chains.

In real applications, knowledge-graphs either do not have type chains, or have incomplete type chains, e.g. Freebase, DBPedia. Automatic extracting fine grained entity typing is an active topic in data-mining and knowledge discovery. Recent progresses are proposed to employ deep neural network architecture integrated with structural and attributive information from DBpedia (Jin et al. 2018). The presented work only uses a single tree structure extracted from knowledge-graph. We are extending current algorithms to address directed acyclic structures. This will lead to a new embedding method for knowledge-graphs.

Another limitation is the scalability problem of the algorithm: When we impose large tree-structures into embedding spaces with zero energy cost, current algorithm will be slow, mainly because of the principle of family action. For example, to impose tree-structured hypernym relations containing 54,310 nodes into the Glove embeddings (Pennington, Socher, and Manning 2014), our current algorithm will take around 6.5 hours.

In the terminology of deep-learning, we developed geometric construction methods to embed tree structures into vector spaces with the zero energy cost, which cannot be achieved by the back-propagation method. Think of these n-balls as a kind of Venn diagrams (Venn 1880) in the embedding space, our method paves paths for integrating logical structured common-sense reasoning (Hayes 1985; Davis and Marcus 2015) with deep-learning, and serves as a simple computational example of how System 1 (simulated by deep-learning) can synergistically work with System 2 (simulated by logic and rule) (Kahneman 2011) – We need to add some parameters, e.g. the length of radius, layer vector, to extend vectors from deep-learning systems into n-balls, and perfectly encode logic and rules into topological relations among these n-balls.

Acknowledgements

Thanks goes to Zhiyuan Liu for viewing the early drafts, to Dong Li for the proof-reading, and to three anonymous reviewers for their constructive comments. Partial financial supports from NSFC (China) under Grant No. 61472177 and No. 61661146007, and BMBF (Germany) under grant number 01/S17064 are greatly appreciated.

References

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD '08*, 1247–1250. New York, NY, USA: ACM.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 26*. Curran Associates. 2787–2795.

Davis, E., and Marcus, G. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Commun. ACM* 58(9):92–103.

Dong, X.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; and Zhang, W. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'14, 601–610. New York, NY, USA: ACM.

Dong, T. 2008. A Comment on RCC: from RCC to RCC^{++}. *Journal of Philosophical Logic* 37(4):319–352.

Durrett, G., and Klein, D. 2015. A joint model for entity analysis: Coreference, typing, and linking. In *TACL*, 477–490.

Fisher, D. H. 1987. Knowledge acquisition via incremental conceptual clustering. *Mach. Learn.* 2(2):139–172.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. The MIT Press.

Gutmann, M. U., and Hyvärinen, A. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.* 13(1):307–361.

Han, X.; Liu, Z.; and Sun, M. 2016. Joint representation learning of text and knowledge for knowledge graph completion. *CoRR* abs/1611.04125.

Hayes, P. J. 1985. The Second Naïve Physics Manifesto. In Hobbs, J. R., and Moore, R. C., eds., *Formal Theories of the Common-Sense World*. Norwood: Ablex. 71–108.

He, S.; Liu, K.; Ji, G.; and Zhao, J. 2015. Learning to represent knowledge graphs with gaussian embedding. In *CIKM '15*, 623–632. New York, USA: ACM.

Ji, G.; He, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL'2015*, 687–696. Beijing: ACL.

Jin, H.; Hou, L.; Li, J.; and Dong, T. 2018. Attributed and predictive entity embedding for fine-grained entity typing in knowledge bases. In *COLING*, 282–292. ACL.

Kahneman, D. 2011. *Thinking, fast and slow*. Allen Lane, Penguin Books. Nobel laureate in Economics in 2002.

LeCun, Y.; Chopra, S.; Hadsell, R.; Ranzato, M.; and Huang, F. J. 2006. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI'15*, 2181–2187. AAAI Press.

Ling, X.; Singh, S.; and Weld, D. S. 2015. Design challenges for entity linking. In *TACL*, 315–328.

Liu, Y.; Liu, K.; Xu, L.; and Zhao, J. 2014. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *COLING*, 2107–2116.

Manning, C. D.; Raghavan, P.; and Schütze, H. 2008. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

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

Nickel, M., and Kiela, D. 2017. Poincaré embeddings for learning hierarchical representations. *CoRR* abs/1705.08039.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP'14*, 1532–1543.

Ren, X.; He, W.; Qu, M.; Huang, L.; Ji, H.; and Han, J. 2016. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *EMNLP*, 1369–1378.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1988. Learning representations by back-propagating errors. In Anderson, J. A., and Rosenfeld, E., eds., *Neurocomputing: Foundations of Research*. Cambridge, MA, USA: MIT Press. 696–699.

Shawe-Taylor, J., and Cristianini, N. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013. Reasoning with neural tensor networks for knowledge base completion. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc. 926–934.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: A core of semantic knowledge. In *WWW '07*, 697–706. New York, NY, USA: ACM.

Venn, J. 1880. On the diagrammatic and mechanical representation of propositions and reasonings. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science* 10(58):1–18.

Wang, Z., and Li, J. 2016. Text-enhanced representation learning for knowledge graph. In *IJCAI*, 1293–1299.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014a. Knowledge graph and text jointly embedding. In *EMNLP*, 1591–1601.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014b. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 1112–1119. AAAI Press.

Xiao, H.; Huang, M.; Hao, Y.; and Zhu, X. 2015a. Transa: An adaptive approach for knowledge graph embedding. *CoRR* abs/1509.05490.

Xiao, H.; Huang, M.; Hao, Y.; and Zhu, X. 2015b. TransG : A generative mixture model for knowledge graph embedding. *CoRR* abs/1509.05488.

Xiao, H.; Huang, M.; and Zhu, X. 2016. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *IJCAI*, 1315–1321.

Xie, R.; Liu, Z.; Jia, J.; Luan, H.; and Sun, M. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*, 2659–2665.

Xu, B.; Zhang, Y.; Liang, J.; Xiao, Y.; Hwang, S.-W.; and Wang, W. 2016. Cross-lingual type inference. In *DASFAA*, 447–462.

Yahya, M.; Berberich, K.; Elbassuoni, S.; and Weikum, G. 2014. Robust question answering over the web of linked data. In *CIKM*, 1107–1116.

Zeng, D.; Liu, K.; Lai, S.; Zhou, G.; and Zhao, J. 2014. Relation classification via convolutional deep neural network. In *COLING*, 2335–2344.

Zhang, D.; Yuan, B.; Wang, D.; and Liu, R. 2015. Joint semantic relevance learning with text data and graph knowledge. In *ACL-IJCNLP*, 32–40.

Zhong, H.; Zhang, J.; Wang, Z.; Wan, H.; and Chen, Z. 2015. Aligning knowledge and text embeddings by entity descriptions. In *EMNLP*, 267–272.