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  
ljjuanzi@tsinghua.edu.cn

Christian Bauckhage,1,2 Armin B. Cremers1  
1B-IT, University of Bonn, Germany  
2Fraunhofer 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 h + r ≈ t, which can be understood as a translation from h to t by adding 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 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 zurich + isa = city and new_york + isa = 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 sub-
space 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 un-
derstood as a recursive usage of supporting vector machines
(SVM) with n-balls as kernels, i.e. (Shawe-Taylor and Cristi-
annini 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 \( t_i \)s and \( h_j \)s, 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 ap-
ply 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.

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 rela-
tions as inclusion relations among \( n \)-balls. Instead of a sin-
gle type, we use fine-grained entity typing in Triple Classifi-
cation; (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; Sec-
tion 4 shows experiment results; Section 5 summarizes the
current work, lists on-going works, and the impact.

Figure 1: \( n \)-ball embedding

Figure 2: \( n \)-ball embedding with subspaces

Related Works

Triple Classification using representational learning is re-
lated to researches in the fields of graph embedding and rea-
soning 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 \( t \approx h + r \)
stand. TransH (Wang et al. 2014b) relaxed this approxima-
tion relation on projection hyperplanes, i.e. \( t \perp \approx \mathbf{d}_r + r \perp \). In
TransR (Lin et al. 2015) and TransD (Ji et al. 2015), the vec-
tor triangle approximation is further updated to \( M_r h + r \approx \)
\( M_t t \), that is, the approximation holds, after the head vec-
tor 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 \( h \) and \( t \) before applying matrix
transformation. The vector triangular approximation holds
for representative vectors within its cluster in the kernel
space, i.e. \( M_r h' + r \approx M_t t' \), where \( h' \) and \( t' \) are the rep-
resentative vectors of \( h \) and \( t \) in their clusters, respectively.
TransA (Xiao et al. 2015a) introduced a weight matrix \( W_r \):
\[
f_r(h, t) = \langle (h + r - t)' \rangle W_r (h + r - t)\rangle.
\]

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 \(M(h, r, t) = D^2_{e}\), 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 \(\delta\) be the margin, \(S\) be the set of true Triples, and \(S'\) be the set of negative samples, the loss-function is written as follows:

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

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

**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 \(B(O_e, r_{de})\), where \(O_e\) is the vector of the central point of the \(n\)-ball of entity \(e\), and \(r_{de}\) is the radius of this \(n\)-ball. \(O_e\) can be further specified by \((d_e, l_e)\), where \(d_e\) is the unit vector representing the direction of \(O_e\), and \(l_e\) is the length of \(O_e\). We define \(n\)-ball as an open space as follows: \(B(O_e, r_{de}) \triangleq \{p|\|O_e - p\| < r_{de}\}\), in which \(\|x - y\| \triangleq \sqrt{(x - y)^\top (x - y)}\). \(B(O_e, r_{de})\) is the set of vector \(p\) whose Euclidean distance to \(O_e\) is less than \(r_{de}\). That is, \(n\)-balls are open spaces. The complement region of an \(n\)-ball is not introduced (Dong 2008).

**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 \(B(O_h, r_{dh})\), which is also an \(n\)-ball, written as \(B(O_h, r_{dh})\). For each tail \(t\) satisfying \((h, r, t)\), \(B(O_t, r_{dt})\) is part of the \(r\)-subspace \(B(O_h, r_{dh})\). We define two predicates for being disconnected (DC) and being inside (P) relations as follows:

- \(B(O_{e_1}, r_{de_1})\) disconnects from \(B(O_{e_2}, r_{de_2})\), written as \(DC(e_1, e_2)\), if the distance between \(O_{e_1}\) and \(O_{e_2}\) is equal...
to or greater than the sum of \( r_{d_1} \) and \( r_{d_2} \).

\[
\text{DC}(e_1,e_2) \triangleq \|O_{e_1} - O_{e_2}\| \geq r_{d_1} + r_{d_2} \quad (1)
\]

- \( \mathbb{B}(O_{e_1}, r_{d_1}) \) is inside \( \mathbb{B}(O_{e_2}, r_{d_2}) \), written as \( P(e_1, e_2) \), if \( r_{d_2} \) equals to or is greater than the sum of \( r_{d_1} \), and the distance between \( O_{e_1} \) and \( O_{e_2} \), as illustrated in Figure 5.

\[
P(e_1, e_2) \triangleq r_{d_2} \geq \|O_{e_1} - O_{e_2}\| + r_{d_1} \quad (2)
\]

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 \( p \) to its central point.
- Rotation transformation, which keeps the length of the radius \( rd \), and rotates angle \( \alpha \) with the \( i \)-th and the \( j \)-th elements of \( O \).

\(^1\)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/GmdlsNat/bp94nb4all

---

**Algorithm 1:** construct \( n \)-ball embeddings

**Input:** known tails: \((h, r, t_1), \ldots, (h, r, t_M), t_i \in \text{tails};\)

- type chain: \( t \text{Chains} = [h, h_1, \ldots, h_N];\)
- pre-trained entity-embeddings: \( \text{EV} \)

**Output:** \( r \)-subspace of \( n \)-ball of head \( h, \text{ballHr};\)

geometric transformation history: \( \text{tranHis} \)

\[
\text{tranHis} = [{} \]

// initialise \( n \)-balls of \([t_1, \ldots, t_M]\)

**foreach** \( \text{ele} \in [t_1, \ldots, t_M] \) do

| \( \text{bTails[ele]} = \text{init_nb}asse(\text{ele}, \text{EV}) \)

end

// make \( n \)-balls of \([t_1, \ldots, t_M]\)

mutually be disconnected using three geometric transformations

\( \text{bTails, tranHis} = \text{disconnect(bTails, tranHis)} \)

\( \text{ballHr} = \text{init_nb}asse(h, \text{EV}) \)

// for each member in \( \text{bTails}, \) create \( n \)-ball of \( r \)-space of \( h \) to contain, using geometric transformations

**foreach** \( \text{ele} \in \text{bTails} \) do

| \( \text{bHr[ele], tranHis} = \text{contain(ballHr, r, ele, tranHis)} \)

end

while \( \text{tChains} \) not empty do

| \( \text{ele} = \text{pop(tChains, 0)} \)

if \( \text{ele} = = h \) then

| // create the minimal cover of

| \( \text{bHr as the isa-subspace inside h's n-ball} \)

| \( \text{ballHr} = \text{mini_cover_ball(bHr)} \)

else

| \( \text{bTs[ele]} = \text{init_nb}asse(\text{ele}, \text{EV}) \)

| \( \text{bTs[ele], tranHis} = \text{contain(bTs[ele], isa, bTs[ele0], tranHis)} \)

| \( \text{ele0} = \text{ele} \)

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: homothetic 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: \textit{Triple\_predict}(x, h, r, KG, γ, EV): whether the Triple \((h, r, x)\) holds in knowledge-graph KG

\textbf{input}: head \(h\), relation \(r\), entity \(x\), knowledge-graph KG, ratio \(γ\), entity-embeddings EV 

\textbf{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 
\(\text{tails} = \text{get\_all\_known\_tails}(h, r, KG)\);

if \(\text{number\_of}(\text{tails}) > 0\) then 

// get the fine grained type chain of \(h\) in KG 
\(\text{tChains} = \text{get\_fine\_grained\_type\_chain}(h, KG)\)

// construct the model, return
\(n\)-ball of \(h\), (2) the initialisation parameter, the sequence of geometric transformations applied for tails 
\(\text{ballH, tranHis} = \text{enlarge\_radius}(\text{ballH}, γ)\)

// construct \(n\)-ball of \(x\) with tranHis 
\(\text{ballX} = \text{construct\_n\_ball}(x, \text{tranHis}, EV)\)

if \(\text{contains\_by}(\text{ballX}, \text{ballH})\) then

\(\text{return True;};\)

else 

\(\text{return False;};\)

else 

\(\text{return False;};\)

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 \(M(h, r)\) 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 \textit{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 renamed them in term of the containment relation of \(n\)-balls as follows.

<table>
<thead>
<tr>
<th>id</th>
<th>WN18/WN11/FB13 relation</th>
<th>(n)-ball relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A B _member_of_domain_topic</td>
<td>contains(_p)(A, B)</td>
</tr>
<tr>
<td>1</td>
<td>A _domain_topic</td>
<td>tr_contains(_p)(A, B)</td>
</tr>
<tr>
<td>2</td>
<td>A _member_meronym</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>3</td>
<td>A _domain_region</td>
<td>tr_contains(_p)(B, A)</td>
</tr>
<tr>
<td>4</td>
<td>A _type_of</td>
<td>tr_contains(_p)(A, B)</td>
</tr>
<tr>
<td>5</td>
<td>A _instance_hypernym</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>6</td>
<td>A _member_holonym</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>7</td>
<td>A _instance_hypernym</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>8</td>
<td>A _has_instance</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>9</td>
<td>A _member_domain_usage</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>10</td>
<td>A _has_part</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>11</td>
<td>A _part_of</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>12</td>
<td>A _synset_domain_topic</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>13</td>
<td>A _synset_domain_topic</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>14</td>
<td>A _gender _of</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>15</td>
<td>A _nationality _of</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>16</td>
<td>A _profession _of</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>17</td>
<td>A _place_of_death</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>18</td>
<td>A _place_of_birth</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>19</td>
<td>A _location</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>20</td>
<td>A _institution</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>21</td>
<td>A _cause_of_death</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>22</td>
<td>A _religion</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>23</td>
<td>A _parents</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>24</td>
<td>A _children</td>
<td>contains(_p)(B, A)</td>
</tr>
<tr>
<td>25</td>
<td>A _ethnicity</td>
<td>contains(_p)(B, A)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#R</th>
<th>#E</th>
<th>#train Triple</th>
<th>#test Triple</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB13-(n)-ball</td>
<td>13</td>
<td>75,043</td>
<td>306,747</td>
<td>45,897</td>
</tr>
<tr>
<td>WN11-(n)-ball</td>
<td>5</td>
<td>38,696</td>
<td>94,472</td>
<td>14,587</td>
</tr>
<tr>
<td>WN18-(n)-ball</td>
<td>7</td>
<td>40,943</td>
<td>60,490</td>
<td>13,148</td>
</tr>
</tbody>
</table>

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 \(i\)th non-symmetric relation, Triple \((h, r, t)\) is named as \textit{contain}_i(h, t) in \(n\)-ball representation, the prefix \textit{tr}_\(r\) is added to \textit{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 hyperparameters, 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 \(M(h, r)\) with \(h\)'s hypernym path and tails of \(h\) in the training set, and record the transformation sequence. Given a knowledge-graph \(KG\), we used entity-embeddings trained in two different ways: (1) only \(KG\), i.e. TransE (Bordes et al. 2013), and (2) \(KG\) with text corpus, i.e. TEKE (Wang and Li 2016).

**Evaluation Protocol.** In transformation-based approach, a threshold \(\delta_t\) is defined to evaluate a score function \(f\): if a transformation score \(f\) of a Triple is below \(\delta_t\), 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 \(x\), \(B(O_x, rd_x)\), the same way as initializing other \(n\)-balls, and transform \(B(O_x, rd_x)\) 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 \(B(O_x, rd_x)\) is located in the \(r\)-subspace of the \(n\)-ball of \(h\), \(B(O_{h_n}, rd_{h_n})\), \((h, r, x)\) will be predicted as true, otherwise false. We update the final radius \(rd_{h_n}\) with a ratio \(\gamma\) 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: Tripes \((h, r, x)\) 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\(^3\).

**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 \(\gamma\) from 0.6 to 2.3, to see whether smaller \(\gamma\) can contribute to precision. Experiment results support that smaller \(\gamma\) 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.

\(^3\)https://figshare.com/articles/FB13nball_zip/7294295
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.

Experiments with WN18-$n$ball Triples in WN18-$n$ball dataset have richer hyponym relations than Triples in 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 $\gamma = 1$.

WN11-$n$ball datasets and pre-trained entity-embeddings are free for downloads.

---

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

4 https://figshare.com/articles/WN18nball/7294316
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, 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. 6166146007, and BMBF (Germany) under grant number 01/S17064 are greatly appreciated.

References


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


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.


