Translating Embeddings for Modeling Multi-relational Data

배지환

Table of Contents

01 Background

02 Introduction

03 Model

04 Experiment

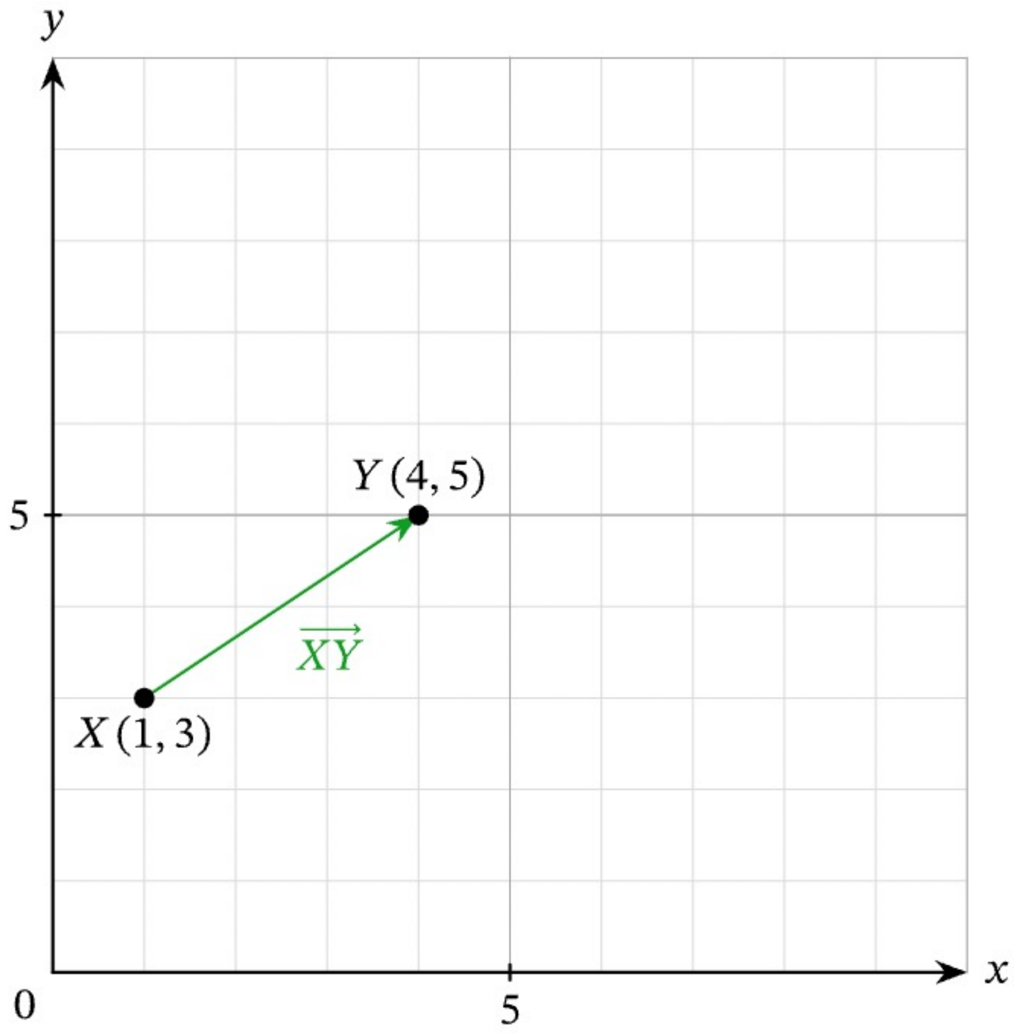
05 Implementation

06 Conclusion

Background

Translating Embeddings for Modeling Multi-relational Data

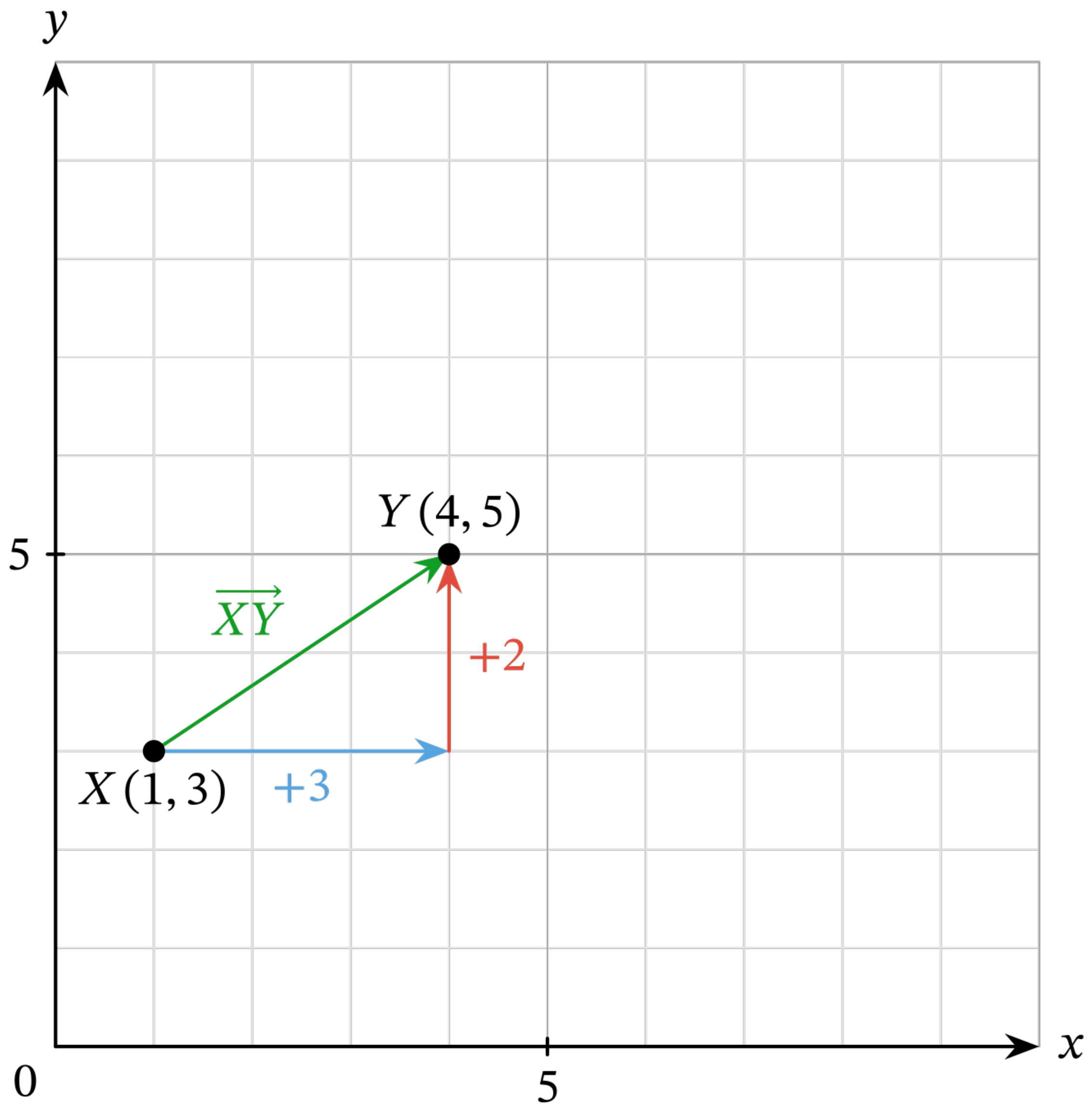
- Background



What is Translation?

> Type of transformation that takes each point in a figure and slides it the same distance in the same direction.

- Background



What is Translation?

“ $(x,y) \to (x+3, y+2)$

Background

Translating Embeddings for Modeling Multi-relational Data

Word Embedding

	1 Man	2 Woman	3 King	4 Queen
Man	1	0	0	0
Woman	0	1	0	0
King	0	0	1	0
Queen	0	0	0	1

Word Embedding

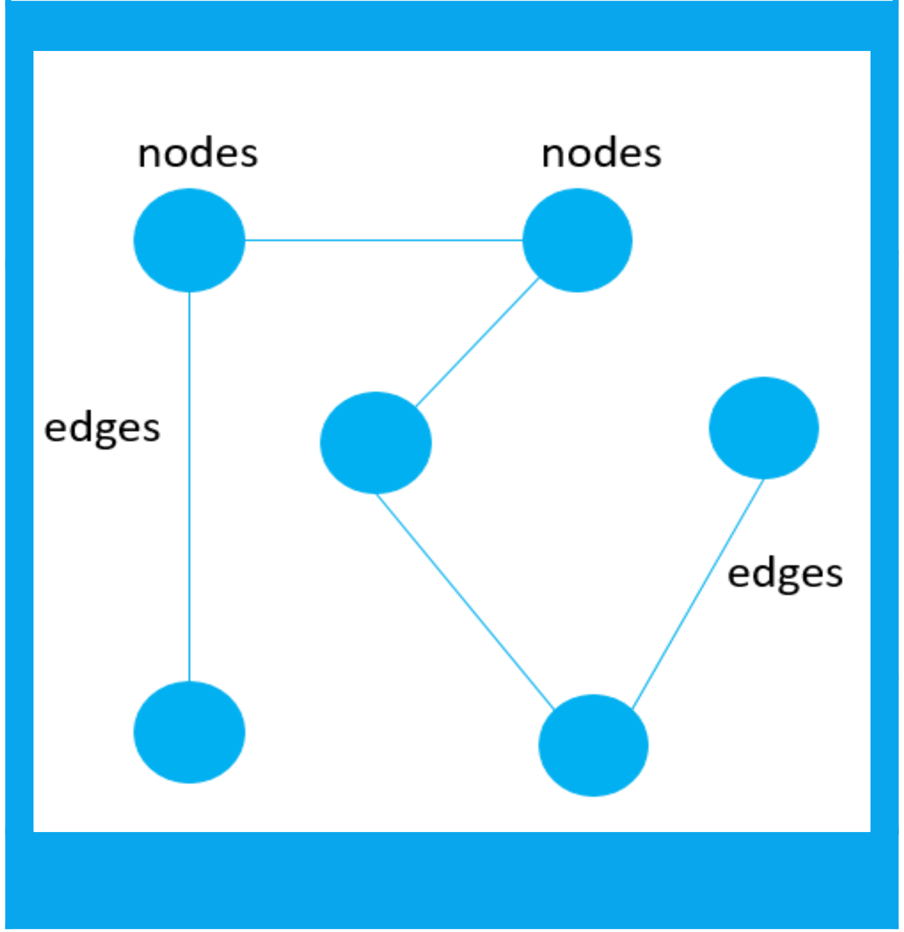
	1 Royalty	2 Femineity
Man	0	0
Woman	0	1
King	1	0
Queen	1	1

Background

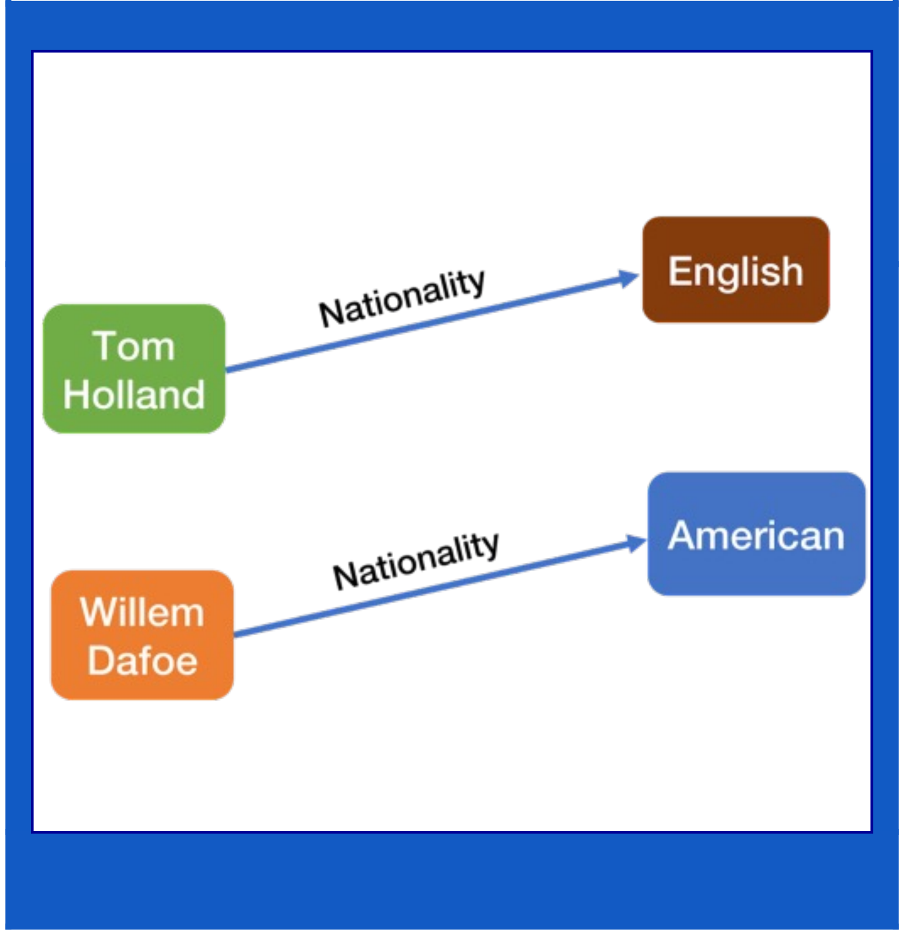
Translating Embeddings for Modeling Multi-relational Data

Multi-Relational Data

Classic Graph Structure



Multi-Relational Structure



Introduction

Set of triplet

Given a training set S of triplets (h, l, t) composed of two entities h, t ∈ E (the set of entities) and a relationship l ∈ L (the set of relationships), our model learns vector embeddings (value in R^k) of the entities and the relationships.

$$S'_{(h,\ell,t)} = \{(h',\ell,t)|h' \in E\} \cup \{(h,\ell,t')|t' \in E\}$$

Corrupted Set

h + l ≈ t

Algorithm
Of TransE

Algorithm 1 Learning TransE

input Training set $S = \{(h, \ell, t)\}$, entities and rel. sets E and L, margin γ, embeddings dim. k.

initialize $\boldsymbol{\ell} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $\ell \in L$

$\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} / \|\boldsymbol{\ell}\|$ for each $\ell \in L$

$\mathbf{e} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$

loop

$\mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity $e \in E$

$S_{batch} \leftarrow \text{sample}(S, b)$ // sample a minibatch of size b

$T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets

for $(h, \ell, t) \in S_{batch}$ **do**

$(h', \ell, t') \leftarrow \text{sample}(S'_{(h,\ell,t)})$ // sample a corrupted triplet

$T_{batch} \leftarrow T_{batch} \cup \{((h, \ell, t), (h', \ell, t'))\}$

end for

Update embeddings w.r.t. $$\boxed{\sum_{\left((h,\ell,t),(h',\ell,t')\right) \in T_{batch}} \nabla \left[\gamma + d(\boldsymbol{h} + \boldsymbol{\ell}, \boldsymbol{t}) - d(\boldsymbol{h'} + \boldsymbol{\ell}, \boldsymbol{t'})\right]_+}$$

end loop

Loss Function

01

$$Loss = \sum_{(h,r,t)\in S} \boxed{d}(h + r, t)$$

dissimilarity function
∈ {L1 norm, L2 norm}

02

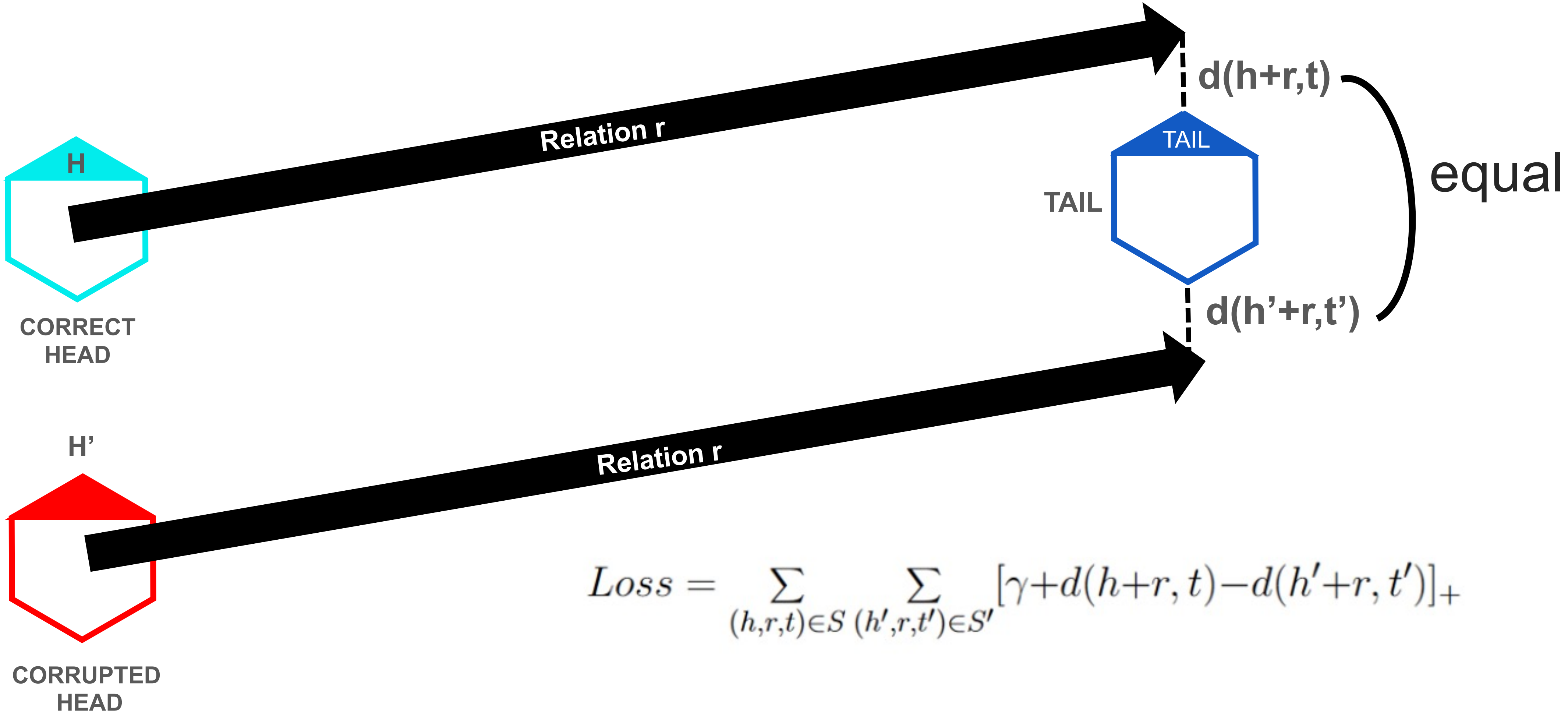
$$Loss = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} [d(h+r,t) - d(h'+r,t')]_+$$

03

$$Loss = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} [\gamma + d(h+r,t) - d(h'+r,t')]_+$$

Why is the margin term included?

$$Loss = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} [d(h+r,t) - d(h'+r,t')]_+$$

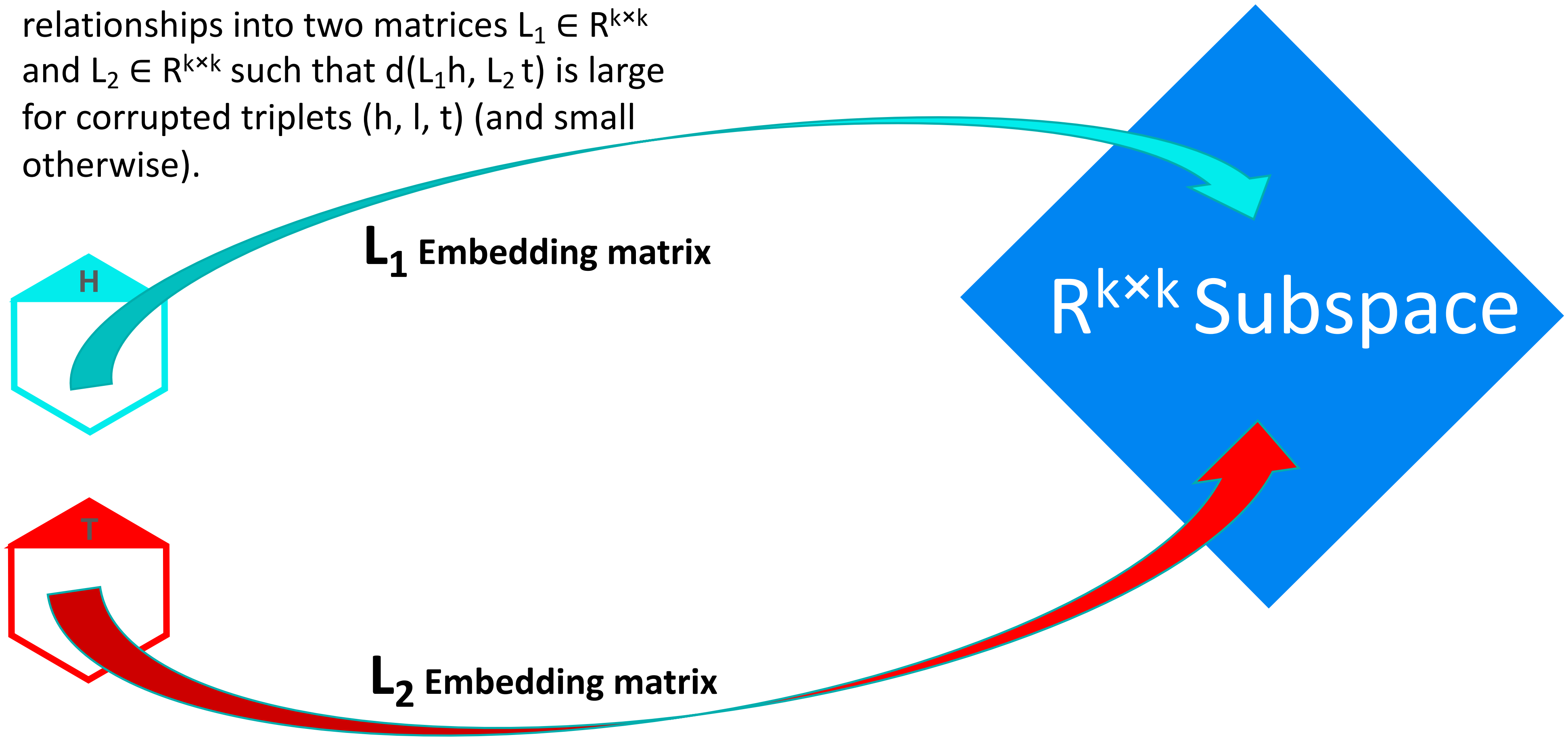


$$Loss = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} [\gamma + d(h+r,t) - d(h'+r,t')]_+$$

Related Works

Structured Embeddings

SE embeds entities into R^k, and relationships into two matrices $L_1 \in R^{k\times k}$ and $L_2 \in R^{k\times k}$ such that $d(L_1 h, L_2 t)$ is large for corrupted triplets (h, l, t) (and small otherwise).



Affine Transformation

“ $d(L_1 h, L_2 t)$ VS $d(h + l, t)$

SE TransE

$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x+1 \\ y \end{bmatrix}$$

Affine Transformation

$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

“ Not Feasible!

Affine Transformation

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+1 \\ y \\ 1 \end{bmatrix}$$

“ Translated by increasing dimension

Related Works

When $\mathbf{L_1}$ reproduce translation and $\mathbf{L_2}$ **= Identity Matrix,**

Structured Embeddings = TransE

Structured Embedding has greater expressiveness than TransE

Neural Tensor Model

$s(h, \ell, t) = \boldsymbol{h}^T \boldsymbol{L} \boldsymbol{t} + \boldsymbol{\ell}_1^T \boldsymbol{h} + \boldsymbol{\ell}_2^T \boldsymbol{t}$, where L ∈ $R^{k \times k}$, L_1 ∈ R^k and L^2 ∈ R^k

'Learning score' for special case of this model

Score is lower scores for corrupted triplet

If we consider TransE with the squared Euclidean distance as dissimilarity function, we have:

$$d(\boldsymbol{h}+\boldsymbol{\ell}, \boldsymbol{t}) = \|\boldsymbol{h}\|_2^2 + \|\boldsymbol{t}\|_2^2 + \|\boldsymbol{\ell}\|_2^2 - 2\left(\boldsymbol{h}^T \boldsymbol{t} + \boldsymbol{\ell}^T (\boldsymbol{t} - \boldsymbol{h})\right)$$

$\|\boldsymbol{h}\|_2^2 = \|\boldsymbol{t}\|_2^2 = 1$

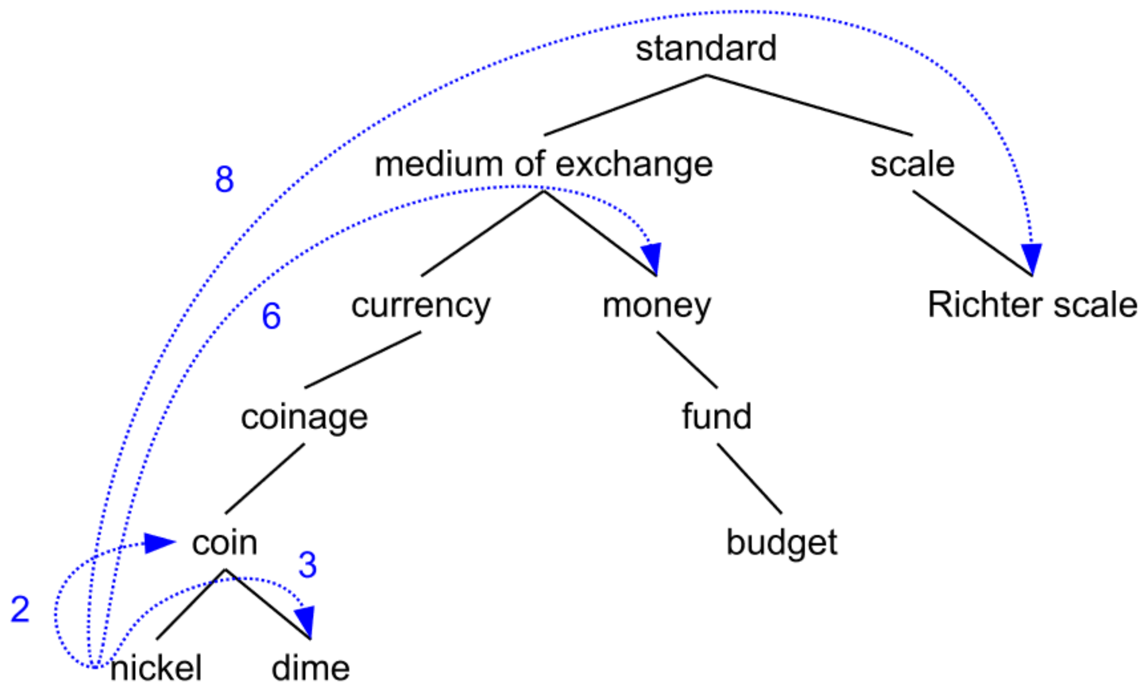
(Due to normalization)

does not play any role in comparing corrupted triplets

Neural Tensor Model Score,
where L is the identity matrix, and $l = l_1 = l_2$

Experiments

Data sets



Wordnet

Freebase

Experiments

Evaluation protocol

Hits@K Hits@K denotes the ratio of the test triples that have been ranked among the top *k* triples, i.e.,

$$\text{Hits@k} = \frac{|\{t \in \mathcal{K}_{test} \mid rank(t) \leq k\}|}{|\mathcal{K}_{test}|}$$

Larger values indicate better performance.

Experiments

Baselines

Unstructured, RESCAL, SE, SME(linear)/SME(bilinear) and LFM

Link Prediction Result

DATASET	WN				FB15K				FB1M	
METRIC	MEAN RANK		HITS@10 (%)		MEAN RANK		HITS@10 (%)		MEAN RANK	HITS@10 (%)
Eval. setting	*Raw*	*Filt.*	*Raw*	*Filt.*	*Raw*	*Filt.*	*Raw*	*Filt.*	*Raw*	*Raw*
Unstructured [2]	315	304	35.3	38.2	1,074	979	4.5	6.3	15,139	2.9
RESCAL [11]	1,180	1,163	37.2	52.8	828	683	28.4	44.1	-	-
SE [3]	1,011	985	68.5	80.5	273	162	28.8	39.8	22,044	17.5
SME(LINEAR) [2]	545	533	65.1	74.1	274	154	30.7	40.8	-	-
SME(BILINEAR) [2]	526	509	54.7	61.3	284	158	31.3	41.3	-	-
LFM [6]	469	456	71.4	81.6	283	164	26.0	33.1	-	-
TransE	**263**	**251**	**75.4**	**89.2**	**243**	**125**	**34.9**	**47.1**	**14,615**	**34.0**

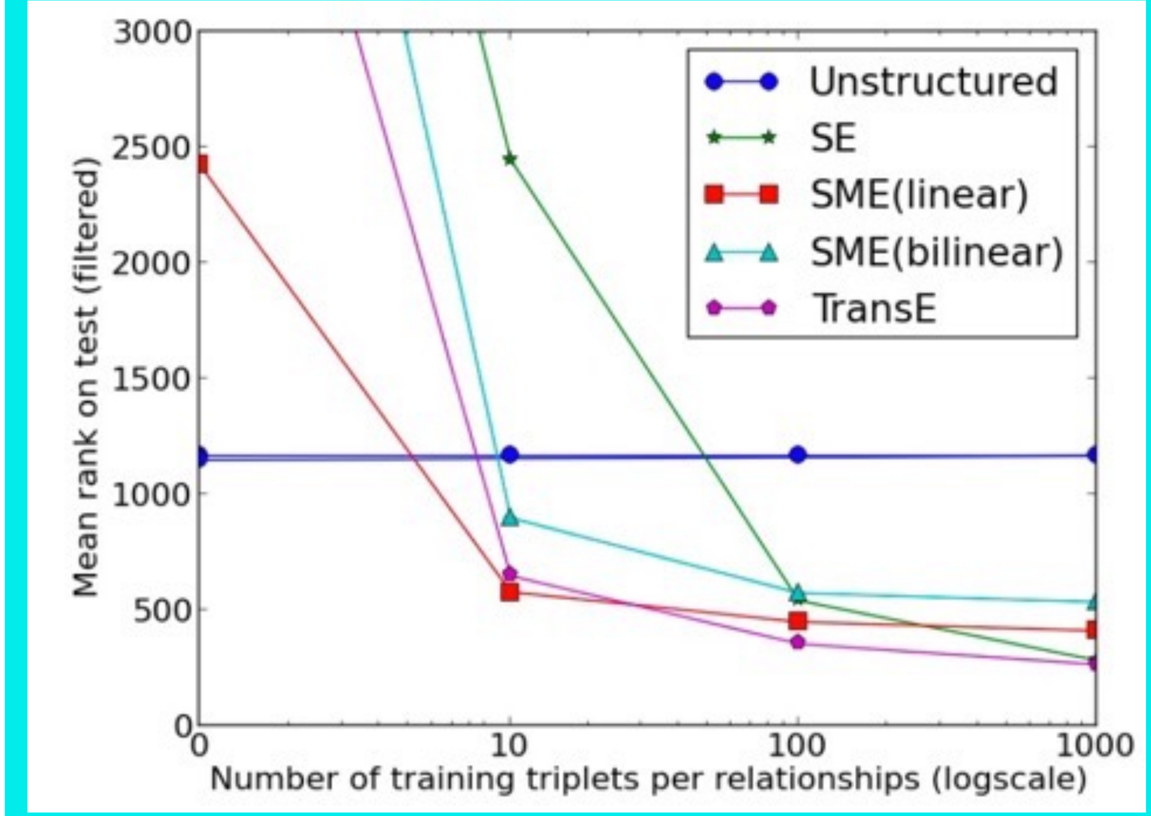
Table 4: **Detailed results by category of relationship.** We compare Hits@10 (in %) on FB15k in the filtered evaluation setting for our model, TransE and baselines. (M. stands for MANY).

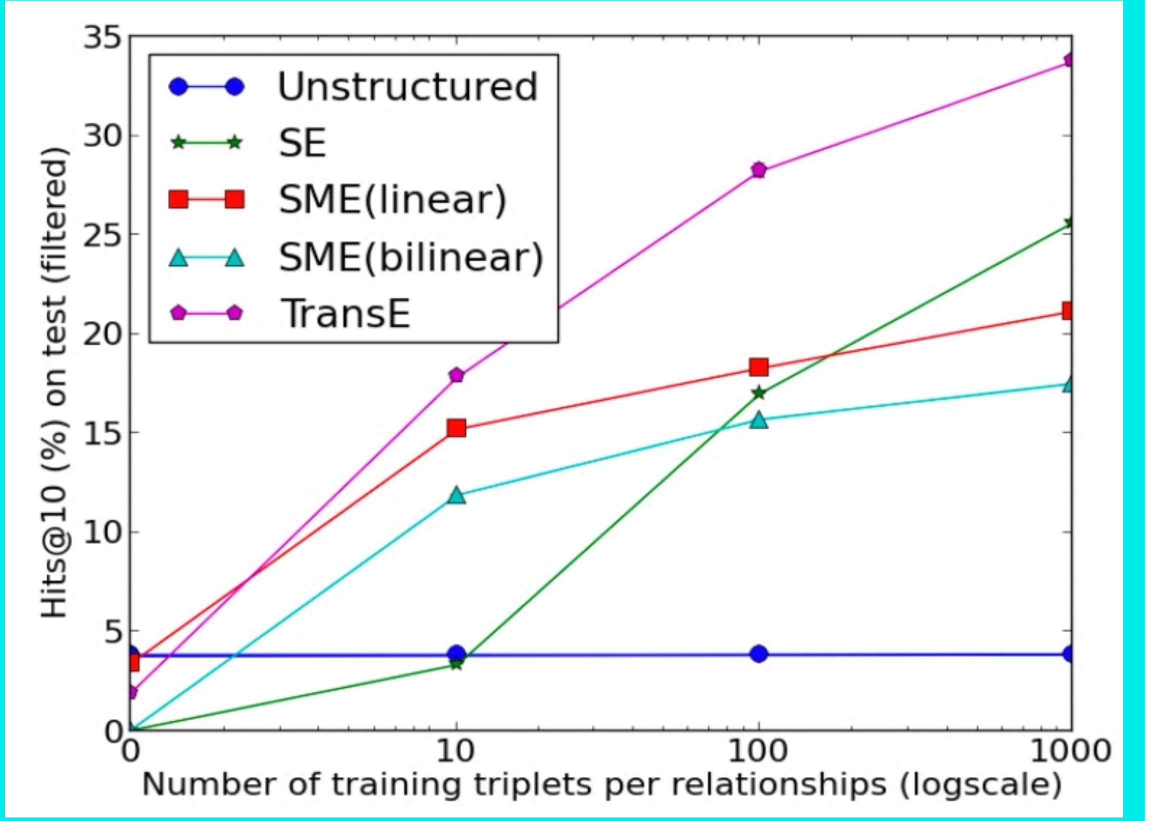
TASK	PREDICTING *head*				PREDICTING *tail*			
REL. CATEGORY	1-TO-1	1-TO-M.	M.-TO-1	M.-TO-M.	1-TO-1	1-TO-M.	M.-TO-1	M.-TO-M.
Unstructured [2]	34.5	2.5	6.1	6.6	34.3	4.2	1.9	6.6
SE [3]	35.6	62.6	17.2	37.5	34.9	14.6	68.3	41.3
SME(LINEAR) [2]	35.1	53.7	19.0	40.3	32.7	14.9	61.6	43.3
SME(BILINEAR) [2]	30.9	**69.6**	**19.9**	38.6	28.2	13.1	**76.0**	41.8
TransE	**43.7**	65.7	18.2	**47.2**	**43.7**	**19.7**	66.7	**50.0**

Table 5: **Example predictions** on the FB15k test set using TransE. **Bold** indicates the test triplet's true tail and *italics* other true tails present in the training set.

INPUT (HEAD AND LABEL)	PREDICTED TAILS
J. K. Rowling `influenced by`	*G. K. Chesterton*, J. R. R. Tolkien, *C. S. Lewis*, **Lloyd Alexander**, Terry Pratchett, Roald Dahl, Jorge Luis Borges, *Stephen King*, Ian Fleming
Anthony LaPaglia `performed in`	*Lantana*, *Summer of Sam*, *Happy Feet*, *The House of Mirth*, Unfaithful, **Legend of the Guardians**, Naked Lunch, X-Men, The Namesake
Camden County `adjoins`	**Burlington County**, *Atlantic County*, *Gloucester County*, Union County, Essex County, New Jersey, Passaic County, Ocean County, Bucks County
The 40-Year-Old Virgin `nominated for`	*MTV Movie Award for Best Comedic Performance*, *BFCA Critics' Choice Award for Best Comedy*, *MTV Movie Award for Best On-Screen Duo*, MTV Movie Award for Best Breakthrough Performance, **MTV Movie Award for Best Movie**, MTV Movie Award for Best Kiss, D. F. Zanuck Producer of the Year Award in Theatrical Motion Pictures, Screen Actors Guild Award for Best Actor - Motion Picture
Costa Rica football team `has position`	*Forward*, *Defender*, *Midfielder*, **Goalkeepers**, Pitchers, Infielder, Outfielder, Center, Defenseman
Lil Wayne `born in`	**New Orleans**, Atlanta, Austin, St. Louis, Toronto, New York City, Wellington, Dallas, Puerto Rico
WALL-E `has the genre`	Animations, Computer Animation, *Comedy film*, *Adventure film*, *Science Fiction*, **Fantasy**, Stop motion, *Satire*, Drama

Learning new relationships with few examples





Implementation of the code

```
print(f"max : {max(head)}")
print(f"length : {len(head)}")
print(f"unique_head : {np.unique(head)}")
print(f"unique_length : {len(np.unique(head))}")
```

```
max : 14540
length : 620232
unique_head : [     0     1     2 ... 14538 14539 14540]
unique_length : 14541
```

unique_head is arithmetic sequence with step size = 1

```
def prepare_neg_entity(head, tail, label):

    rel_matrix = np.zeros((len(np.unique(head)), len(np.unique(head))))
    for h,t,l in zip(head, tail, label):
        rel_matrix[h,t] = 1

    rel_matrix[rel_matrix>0] = -1
    rel_matrix[rel_matrix==0] = 1
    series = torch.arange(np.max(head)+1)
```

rel_matrix is an adjacent matrix filled with relationship ‘l’ if ‘l’ exists for the head and tail pair. Filled with zero otherwise.

Implementation of the code

```
def prepare_neg_entity(head, tail, label):

    rel_matrix = np.zeros((len(np.unique(head)), len(np.unique(head))))
    for h,t,l in zip(head, tail, label):
        rel_matrix[h,t] = 1

    rel_matrix[rel_matrix>0] = -1
    rel_matrix[rel_matrix==0] = 1
    series = torch.arange(np.max(head)+1)

    head_neg = {}
    for h in np.unique(head):
      temp = np.multiply(series, rel_matrix[h, :])
      temp = temp[(temp >= 0)]
      head_neg[h] = deepcopy(temp)

    tail_neg = {}
    for t in np.unique(tail):
      temp = np.multiply(series, rel_matrix[:, t])
      temp = temp[(temp >= 0)]
      tail_neg[t] = deepcopy(temp)
    return head_neg, tail_neg
```

Head and tail pair with relationships (correct pair)

Potential corrupted set

Element-wise multiplication of the series by the relation matrix of the particular head

Only the potential corrupted set pairs left

Implementation of the code

```
#여기서부터 model
class TransE(nn.Module):
    def __init__(self, num_entity, num_label, embed_dim, gamma, configure):
        super().__init__()
        self.embed_label = nn.Embedding(num_label, embed_dim)
        nn.init.uniform_(self.embed_label.weight,
        -6/torch.sqrt(torch.tensor(embed_dim)),
        6/torch.sqrt(torch.tensor(embed_dim)))

        self.embed_entity = nn.Embedding(num_entity, embed_dim)
        nn.init.uniform_(self.embed_entity.weight,
        -6/torch.sqrt(torch.tensor(embed_dim)),
        6/torch.sqrt(torch.tensor(embed_dim)))

        self.gamma = gamma
        self.embed_dim = embed_dim
        self.configure = configure
```

Xavier Initialization

Implementation of the code

```
def forward(self, batch):
    head, label, tail = batch['head'], batch['label'], batch['tail']
    head_p, tail_p = batch['head_p'], batch['tail_p']
    batch_size = head.size(0)

    head = self.embed_entity(head) # (batch_size, embed_dim)
    tail = self.embed_entity(tail) # (batch_size, embed_dim)
    rel = self.embed_label(label) # (batch_size, embed_dim)

    head_prime  = torch.tensor(head_p).to(self.configure.device).long()
    #to match the input data type
    tail_prime  = torch.tensor(tail_p).to(self.configure.device).long()
    #to match the input data type

    head_prime = self.embed_entity(head_prime)
    # (batch_size, neg_sample, embed_dim)
    tail_prime = self.embed_entity(tail_prime)
    # (batch_size, neg_sample, embed_dim)

    dsm_correct = torch.norm(head + rel - tail, 2, dim=1) # (batch_size)
    dsm_corrupt_1 = torch.norm(head_prime + rel.unsqueeze(1)
                            - tail.unsqueeze(1), 2, dim=2)
    # (batch_size, neg_sample)
    dsm_corrupt_2 = torch.norm((head + rel).unsqueeze(1)
                            - tail_prime, 2, dim=2)
    # (batch_size, neg_sample)
    loss = torch.max(torch.sum(0*dsm_correct), torch.sum(self.gamma
            + 2*self.configure.neg_sample*torch.sum(dsm_correct)
            - torch.sum(dsm_corrupt_1 + dsm_corrupt_2)))
    return loss
```

Unsqueezed to match the dimension and calculated norm along the embed_dim axis

max(0,b) function to implement the function of returning only the positives as the paper states :

where $[x]_+$ denotes the positive part of x,

For each positive set, there are 2 corrupted set(h_p, t_p) each with configure.neg_sample # of neg samples. Thus, 2 x #neg_sample

Implementation of the code

```
#여기서부터 dataset
class KgDataset(Dataset):
    def __init__(self, head, tail, label, neg_sample_k, head_neg, tail_neg):
        super().__init__()
        self.head = head
        self.tail = tail
        self.label = label
        self.neg_sample_k = neg_sample_k
        self.head_neg = head_neg
        self.tail_neg = tail_neg

    def __len__(self):
        return len(self.head)

    def __getitem__(self, idx):
        head, tail, label = self.head[idx], self.tail[idx], self.label[idx]
        tail_prime = np.random.choice(self.head_neg[head], self.neg_sample_k)
        head_prime = np.random.choice(self.tail_neg[tail], self.neg_sample_k)

        return {'head':head, 'label':label, 'tail':tail, 'tail_p':tail_prime, 'head_p':head_prime}
```

Implementation of the code

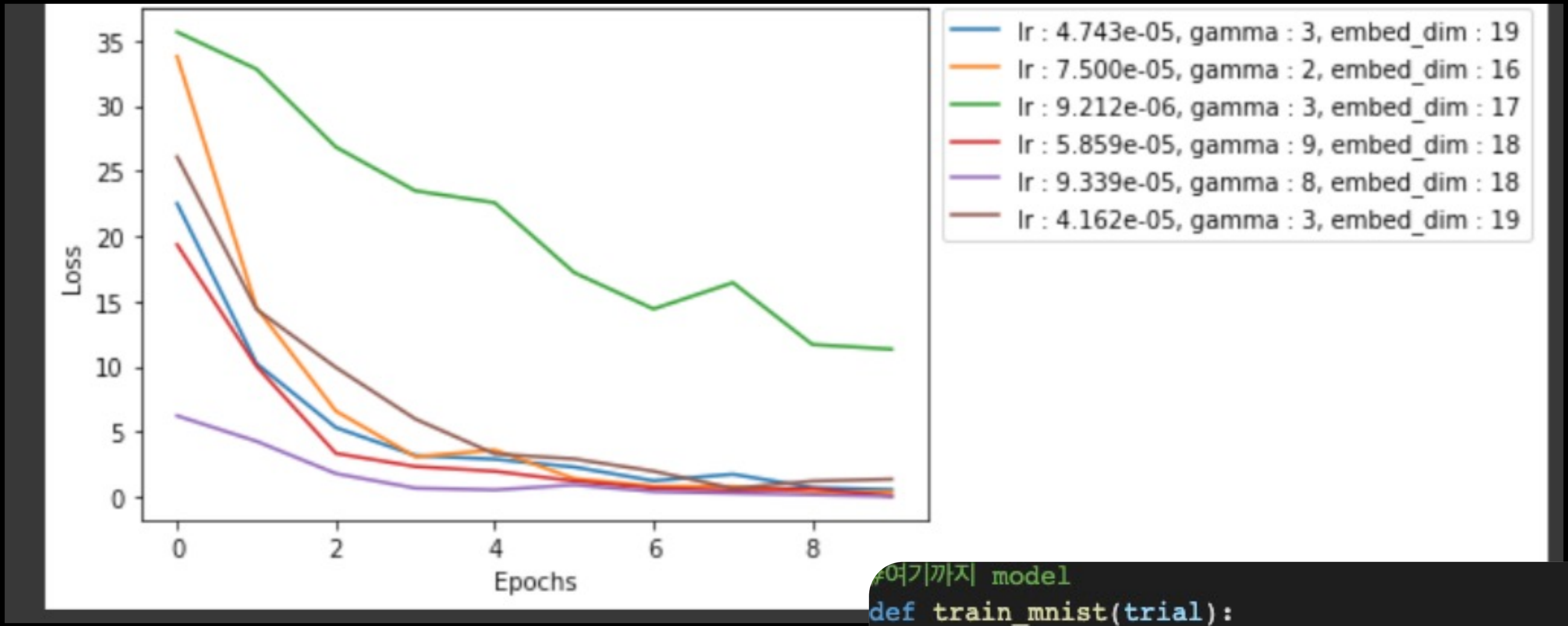
```
  for epoch in range(configure.epochs):
      losses = []
      model.train()
      for batch_data in dataloader:
          optimizer.zero_grad()
          batch_data = {k:v.to(configure.device) for k,v in batch_data.items()}
          loss = model(batch_data)
          losses.append(loss.item())
          loss.backward()
          optimizer.step()
      print(f'EPOCH {epoch+1} : Loss {np.mean(losses):.1f}')
      history['train'].append(np.mean(losses))
  plt.plot(history['train'], label =
  f'lr : {np.format_float_scientific(configure.learning_rate, unique=False, precision=3)}, gamma : {configure.gamma}, embed_dim : {configure.embed_dim}')
  plt.xlabel('Epochs')
  plt.ylabel('Loss')
  plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
  return losses[-1]


sampler = optuna.samplers.TPESampler()

study = optuna.create_study(direction="minimize")
study.optimize(train_mnist, n_trials=6)
df = study.trials_dataframe()
df.head(3)

trial = study.best_trial
print('Accuracy: {}'.format(trial.value))
print("Best hyperparameters: {}".format(trial.params))
```

Hyperparameter Tuning




```
#여기까지 model
def train_mnist(trial):

  class Configure():
          learning_rate = trial.suggest_uniform('learning_rate', 1e-6, 1e-4)
          batch_size = 1024
          embed_dim = int(trial.suggest_uniform('embed_dim', low = 15, high = 20))
          device = 'cuda:0' if torch.cuda.is_available() else 'cpu'
          neg_sample = 5 # h'와 t' 각각 count
          gamma = int(trial.suggest_uniform('gamma', low= 1, high= 10))
          epochs = 10
```

Summary&Conclusion

TransE is an approach to learn embeddings of KBs, focusing on the minimal parametrization of the model to primarily represent hierarchical relationships.

TransE is highly scalable model, as shown through the application of a very large-scale chunk of data

Even in complex and heterogeneous multi-relational domains simple yet appropriate modeling assumptions can lead to better trade-offs between accuracy and scalability.

The greater expressivity of these models comes at the expense of substantial increases in model complexity which results in modeling assumptions that are hard to interpret, and in higher computational costs.

THANK YOU