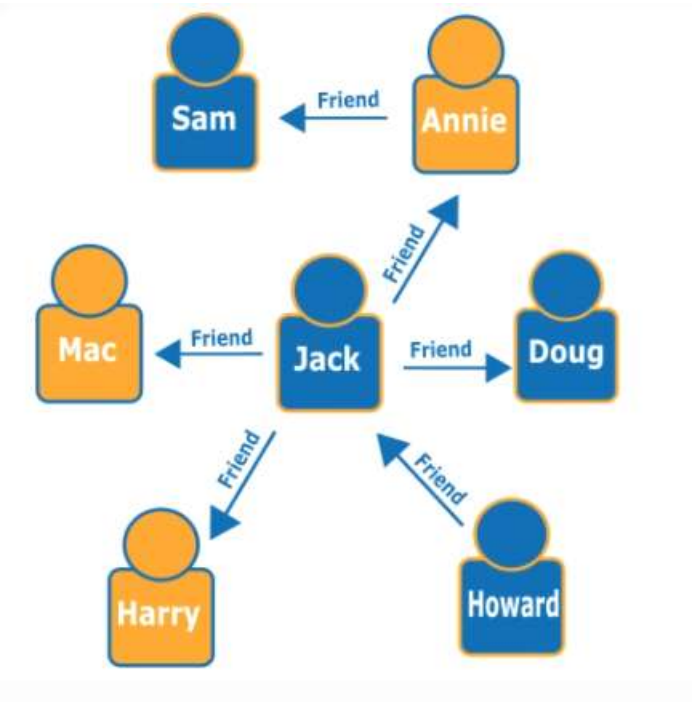
Translating Embeddings for Modeling Multi-relational Data

최서웅

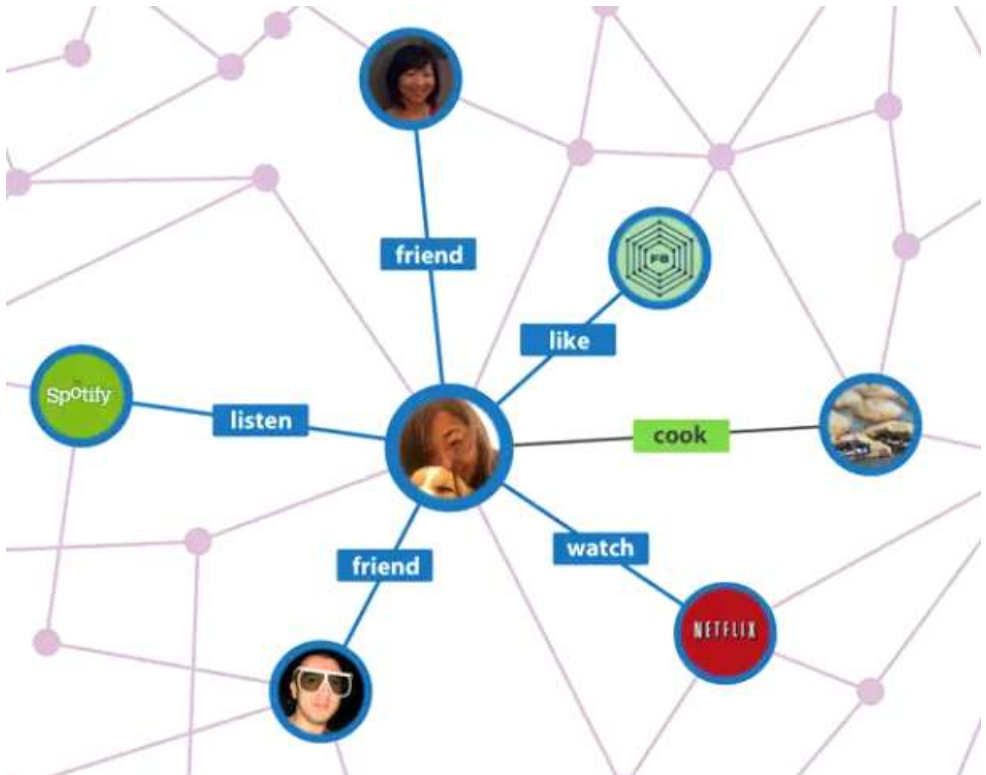
Index

Background

Single-Relational Data: At most one type of relationship



Multi-Relational Data : More than one type of relationship



Background

Knowledge Base : 특정 분야와 관련된 지적 활동과 경험을 통해서 축적한 전문 지식 그리고 문제 해결에 필요한 사실과 규칙 등이 저장되어 있는 데이터베이스

Entity : abstract concepts or concrete entity of the world

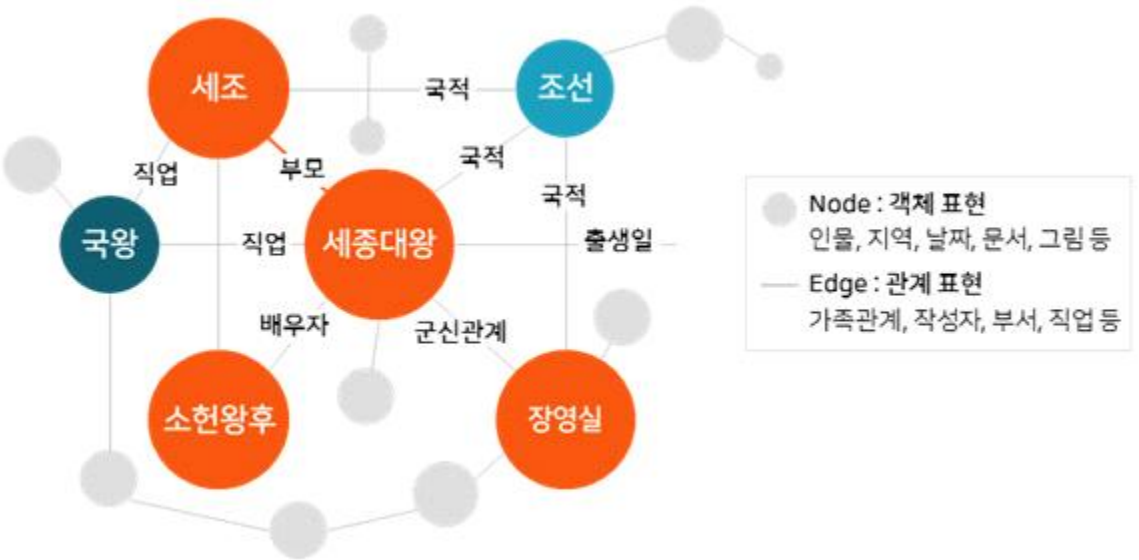
Relationship: predicates that represent facts involving two of them

Background

Knowledge Graph : 지식 베이스로 만들어진 그래프로 KB를 특정한 방식으로 구성하여 데이터 간의 관계를 명확하게 매핑한다.

Node : Entity

Edge : Relationship

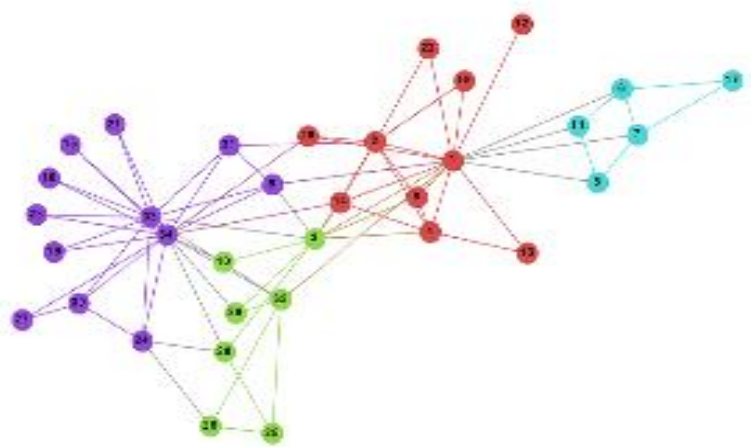


Background

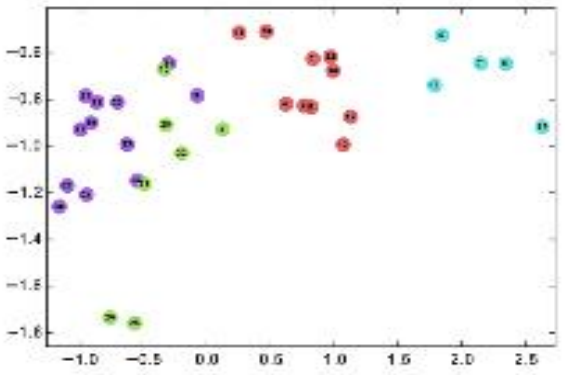
Embedding : representation of a data item (like a word, sentence, user, item, entity etc.) as a vector of real numbers.

Learned Embedding

	embedding pairs			
request →	0.007	-0.039	-0.002	-0.024
feature →	-0.014	-0.011	0.014	-0.017
how →	0.01	0.004	0.006	-0.037
issue →	-0.015	-0.035	-0.009	-0.01
credit →	-0.012	-0.027	-0.001	-0.05



(a) Input: Karate Graph



(b) Output: Representation

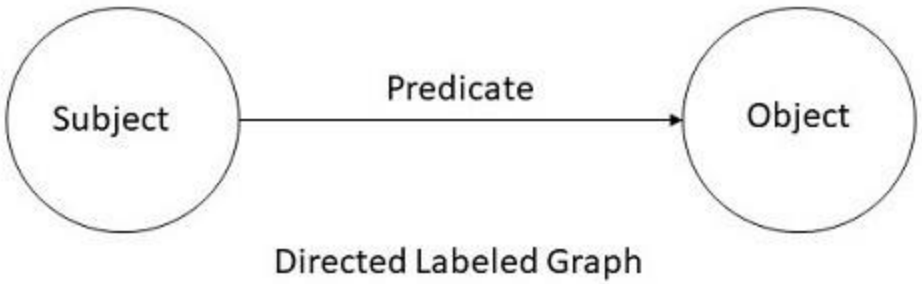
Introduction

For multi-relational data…

In Directed Graph

Node : Entity

Edge : Relation, (h, l, t) = (head, label, tail)



Focus on modeling multi-relational data in a knowledge base.

Goal : Automatically add new facts to complete the knowledge base, without the need for additional knowledge

Introduction

Modeling multi-relational data

Extract local or global connectivity pattern. Generalize relationships between entities using pattern.

- In multi-relational data, locality involve relationships and entities of different types at the same time
  - There can be relationships and entities across various heterogeneous data, a more generalized method is required, not simply a descriptive analysis (like 'a friend of a friend is a friend').
- So requires more generic approach

Introduction

Modeling multi-relational data

Recent approach : Expressivity, Universality

Expressivity ↑ Complexity, Computation ↑

High capacity → Hard to Regularization, Overfitting

Non-convex → local-minima , underfitting

simple yet appropriate modeling assumptions can lead to better trade-offs between accuracy and scalability!

Introduction

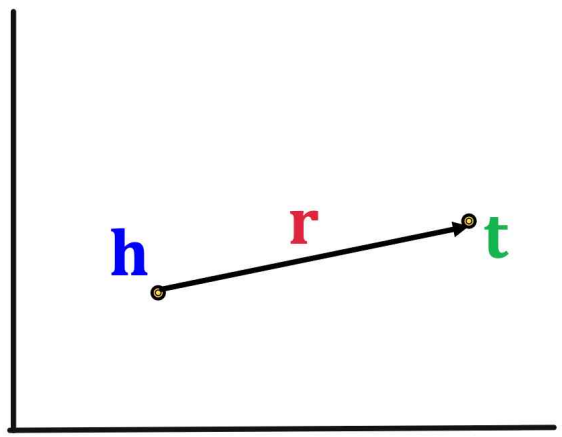
Relationships as translations in the embedding space

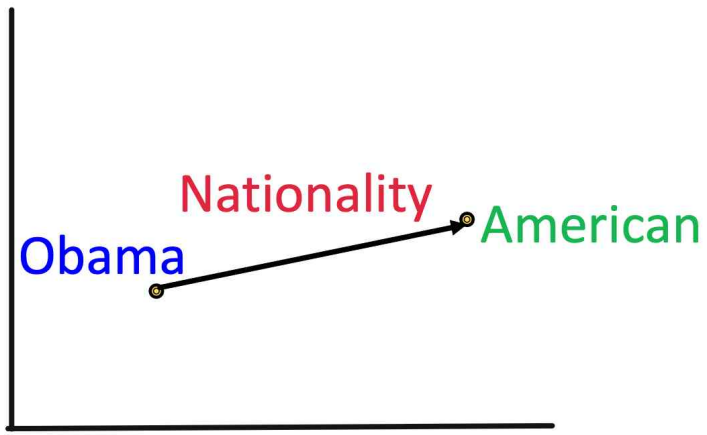
TransE : Energy-based model to learn low-dimensional embedding of entities

Relationship = translations in embedding space

If (h,r,t) holds, then h+r should be closed to t

$$h + r = t$$





Introduction

Why use translation-based parametrization?

1. Hierarchical relations are common in knowledge base, and translation is a natural method for representing them.
2. Can represent one-to-one relationship of entities of different types

Works powerfully in most types of relationships

Performs well in link prediction in real-world knowledge bases

Light parameterization

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

  $\ell \leftarrow \ell / \|\ell\|$ 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. $\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla \left[\gamma + d(\boldsymbol{h} + \boldsymbol{\ell}, \boldsymbol{t}) - d(\boldsymbol{h}' + \boldsymbol{\ell}, \boldsymbol{t}')\right]_+$

end loop

Set of triplets :$S = \{(h, l, t)\}$

Entities Set: E

Relationships Set : L

Margin : γ

Embedding dim : k

TransE

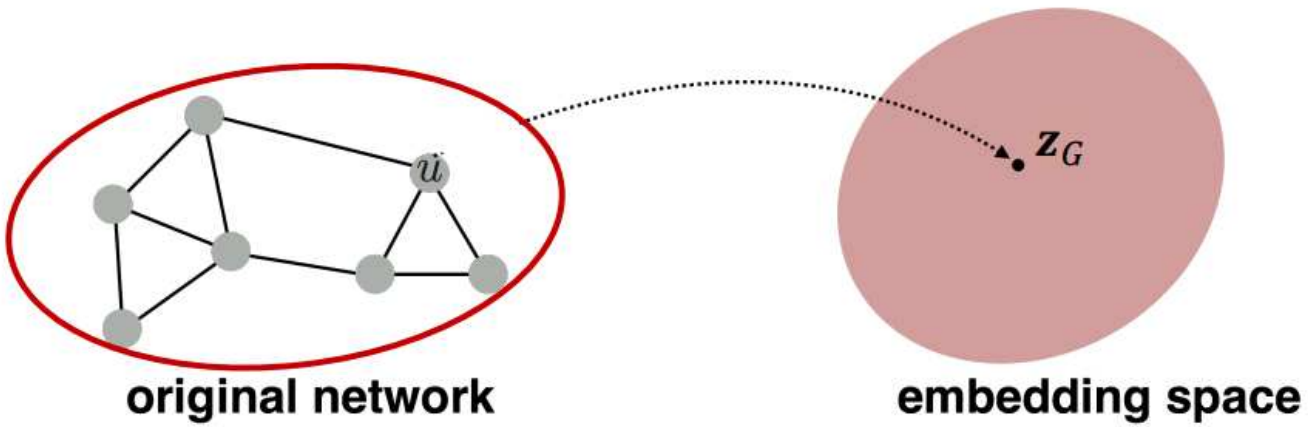
Learn vector embeddings of entity and relationships

Vector takes value in $\mathbb{R}^k$

If (h, l, t) holds, $h + l \approx t$

d : Dissimilarity measure (L1-norm, L2-norm)

$d(h + l, t)$: Energy of triplet



TransE

$$\mathcal{L} = \sum_{(h,\ell,t)\in S} \sum_{(h',\ell,t')\in S'_{(h,\ell,t)}} \left[\gamma + d(\boldsymbol{h}+\boldsymbol{\ell},\boldsymbol{t}) - d(\boldsymbol{h'}+\boldsymbol{\ell},\boldsymbol{t'})\right]_+ \quad (1)$$

where $[x]_+$ denotes the positive part of x, $\gamma > 0$ is a margin hyperparameter, and

$$S'_{(h,\ell,t)} = \left\{(h',\ell,t)|h' \in E\right\} \cup \left\{(h,\ell,t')|t' \in E\right\}. \quad (2)$$

Loss function

S' : set of corrupted triplets (training triplets with either the head or tail replaced by a random entity (but not both at the same time)

Lower values of the energy for training triplets than for corrupted triplets

$d(h+l,t)\ \downarrow$, $d(h'+l,t')\ \uparrow\ \rightarrow \mathcal{L}\ \downarrow$

TransE

$$\mathcal{L} = \sum_{(h,\ell,t)\in S} \sum_{(h',\ell,t')\in S'_{(h,\ell,t)}} \left[\gamma + d(\boldsymbol{h}+\boldsymbol{\ell},\boldsymbol{t}) - d(\boldsymbol{h'}+\boldsymbol{\ell},\boldsymbol{t'})\right]_+$$

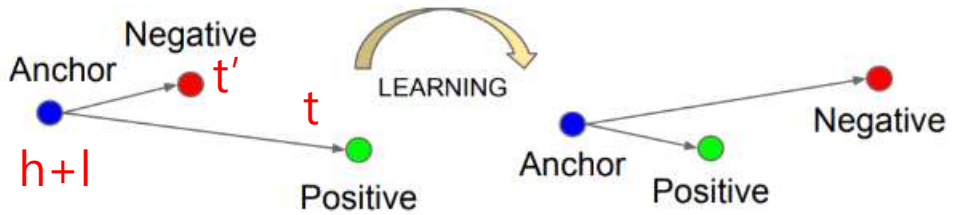


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

$d(h+l,t)+\gamma < d(h'+l,t')$ → Distance between the anchor and the negative is greater than the distance between the anchor and the positive by at least margin

➔Loss only consider cases that do not satisfy this condition

max(0, (margin +d(h+l, t)) – d(h′+l, t′))

$d(h'+l,t') - d(h+l,t) < \gamma$ → Model only consider cases where the distance between the positive and the negative is not large

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

  $\ell \leftarrow \ell / \|\ell\|$ 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. $\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla [\gamma + d(h + \ell, t) - d(h' + \ell, t')]_+$

end loop

Algorithm

Initialize entity and relation embeddings with small random values

For each loop

Normalize the entity embeddings to avoid large magnitudes

Shuffle the triples and generate mini-batches for training.

For each triplets, make the corrupted triplets

For each batch, compute the gradients and update the embeddings.

Related Work

Table 1: **Numbers of parameters** and their values for FB15k (in millions). n_e and n_r are the nb. of entities and relationships; k the embeddings dimension.

METHOD	NB. OF PARAMETERS	ON FB15K
Unstructured [2]	$O(n_e k)$	0.75
RESCAL [11]	$O(n_e k + n_r k^2)$	87.80
SE [3]	$O(n_e k + 2n_r k^2)$	7.47
SME(LINEAR) [2]	$O(n_e k + n_r k + 4k^2)$	0.82
SME(BILINEAR) [2]	$O(n_e k + n_r k + 2k^3)$	1.06
LFM [6]	$O(n_e k + n_r k + 10k^2)$	0.84
TransE	$O(n_e k + n_r k)$	0.81

SE

Entity $\in \mathbb{R}^k$, Relationship $L_1 \in \mathbb{R}^{k \times k}$, $L_2 \in \mathbb{R}^{k \times k}$

Basic Idea : If two entities(h, t) is in same triplet, then embeddings of two entities should be closed in some subspace

Dissimilarity measure : $d(L_1 h, L_2 t)$

Reflect the Assymetry (남한의 수도는 서울, 서울의 수도는 남한이 아님)

$R_k = \left(R_k^{lhs}, R_k^{rhs}\right)$

$L_1 = L, L_2 = I$ → SE = TransE

Better Expressivity than TransE, but lower performance

Related Work

Neural Tensor model

$$s(h, \ell, t) = \boldsymbol{h}^T \boldsymbol{L} \boldsymbol{t} + \boldsymbol{\ell}_1^T \boldsymbol{h} + \boldsymbol{\ell}_2^T \boldsymbol{t}$$

Score of special case : $s(h, l, t)$

$L \in \mathbb{R}^{k \times k}$, $L_1 \in \mathbb{R}^k$, $L_2 \in \mathbb{R}^k$ (low score for corrupted triplet)

Let dissimilarity function of TransE squared Euclidean distance

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

$$\boldsymbol{h}^T \boldsymbol{t} + \boldsymbol{\ell}^T (\boldsymbol{t} - \boldsymbol{h})$$

$||h||^2 = ||t||^2 = 1, L = I, l = l_1 = -l_2, ||l||_2^2$ doesn't play any role in comparing corrupted triplet

TransE has less parameters → simple train, prevent underfitting

Experiment

Dataset

Wordnet

Entity = meaning of word, relationships = lexical relation

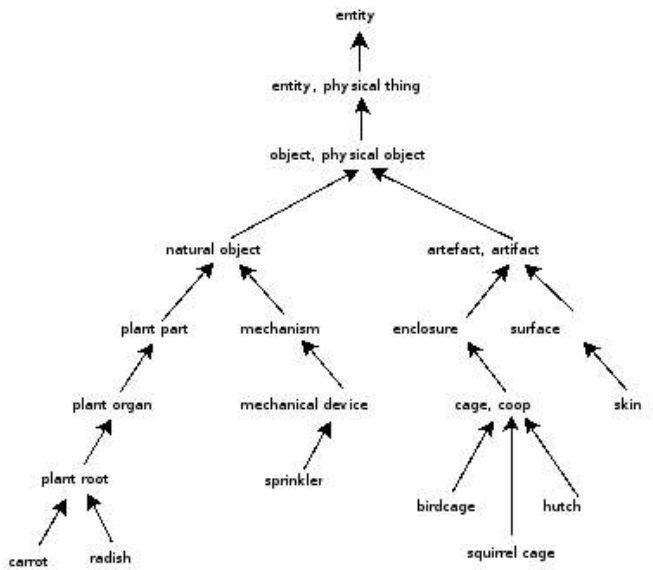


Figure 1. "is a" relation example

Experiment

Freebase

Huge and growing KB of general facts

1. FB15K : present in the Wikilinks databases and that also have at least 100 mentions in Freebase
2. selecting the most frequently occurring 1 million entities

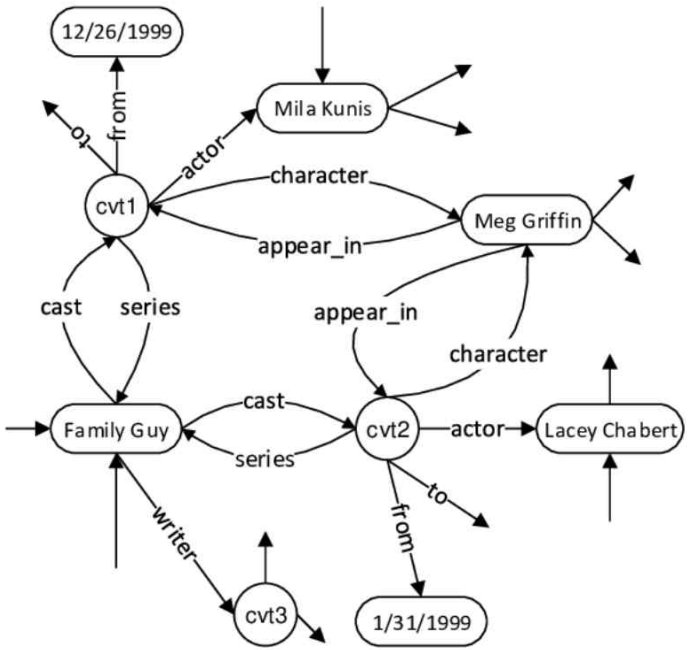


Table 2: **Statistics of the data sets** used in this paper and extracted from the two knowledge bases, Wordnet and Freebase.

DATA SET	WN	FB15K	FB1M
ENTITIES	40,943	14,951	1×10^6
RELATIONSHIPS	18	1,345	23,382
TRAIN. EX.	141,442	483,142	17.5×10^6
VALID EX.	5,000	50,000	50,000
TEST EX.	5,000	59,071	177,404

Experiment

Evaluation metric

Method

1. For each test triplet, replace head with each entities.
2. Compute dissimilarity of corrupted triplets and sort by ascending order.
3. Repeat 1,2 by replacing tail instead of head

Hits@10 : the proportion of correct entities ranked in the top 10

Mean rank : mean of those predicted ranks

Filtered : Remove the corrupted triplets that are in original dataset.

Raw : Do test without removing

Experiment

Baselines

Unstructured : TransE that considers the data as mono-relational and sets all translations to 0

RESCAL : collective matrix factorization model

SE, SME(linear)/SME(bilinear), LFM : Energy-based model

Table 1: **Numbers of parameters** and their values for FB15k (in millions). n_e and n_r are the nb. of entities and relationships; k the embeddings dimension.

METHOD	NB. OF PARAMETERS	ON FB15K
Unstructured [2]	$O(n_e k)$	0.75
RESCAL [11]	$O(n_e k + n_r k^2)$	87.80
SE [3]	$O(n_e k + 2n_r k^2)$	7.47
SME(LINEAR) [2]	$O(n_e k + n_r k + 4k^2)$	0.82
SME(BILINEAR) [2]	$O(n_e k + n_r k + 2k^3)$	1.06
LFM [6]	$O(n_e k + n_r k + 10k^2)$	0.84
TransE	$O(n_e k + n_r k)$	0.81

Experiment

Hyperparameter for TransE

Epochs : 1000

Learning rate : {0.001, 0.01, 0.1}

Margin : {1, 2, 10}

Dimension : {20, 50}

Measure : {L1, L2}

Wordnet : k = 20, Learning Rate = 0.01, Margin = 2, d = L1;

FB15k : k = 50, Learning Rate= 0.01, Margin = 1, d = L1;

FB1M : k = 50, Learning rate= 0.01, Margin = 1, d = L2;

Experiment

Table 3: **Link prediction results.** Test performance of the different methods.

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**

Experiment

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**

Good performance at 1-To-1

Experiment

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

Even if the good answer is not always top-ranked, the predictions reflect common-sense.

Experiment

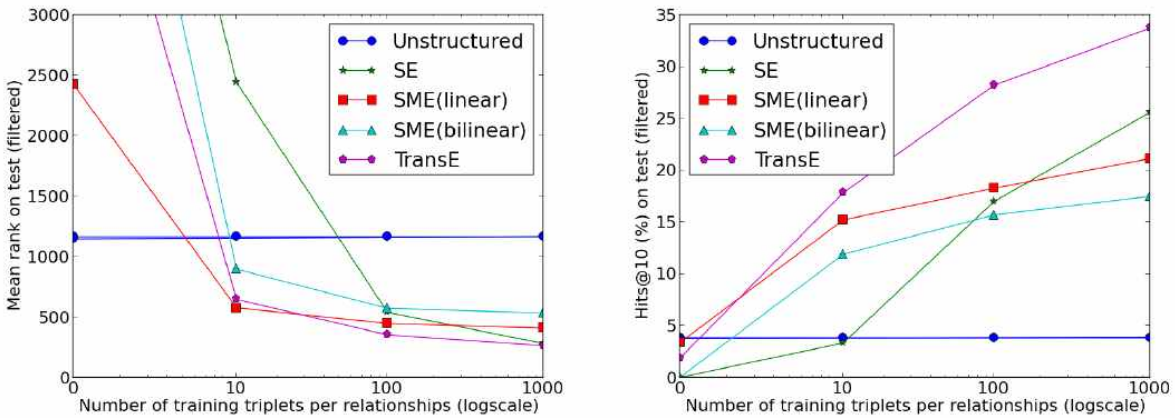


Figure 1: **Learning new relationships with few examples**. Comparative experiments on FB15k data evaluated in mean rank (left) and hits@10 (right). More details in the text.

Select 40 relationships

Split dataset : FB15k-40rel, FB15k-rest (Two datasets involve all entities)

(1) Trained and selected using FB15k-rest training and validation sets

(2) Trained on the training set FB15k-40rel but only to learn the parameters related to the fresh 40 relationships

(3) Evaluated in link prediction on the test set of FB15k-40rel

(4) Repeat this procedure while using 0, 10, 100 and 1000 examples of each relationship.

→ Simplicity of TransE makes it generalize well, without modifying any of the already trained embeddings.

Conclusion

New approach to learn embedding of KB

Minimal parametrization of the model to represent hierarchical relationship

Work very well compared to competing methods

Highly scalable model

Unclear if all relationship types(1-to-1, 1-to-Many etc) can be modeled adequately, performing well compared to other approaches across all settings.

Implementation

Initialized embeddings and normalize relation embeddings

```
# Initialize entity and relation embeddings
self.entity_embeddings = torch.empty(num_entities, embedding_dim, device=self.device)
torch.nn.init.uniform_(self.entity_embeddings, -6/np.sqrt(embedding_dim), 6/np.sqrt(embedding_dim))

self.relation_embeddings = torch.empty(num_relations, embedding_dim, device=self.device)
torch.nn.init.uniform_(self.relation_embeddings, -6/np.sqrt(embedding_dim), 6/np.sqrt(embedding_dim))

# Normalize relation embeddings
self.relation_embeddings = self.relation_embeddings / torch.norm(self.relation_embeddings, p=2, dim=1, keepdim=True)
```

Implementation

Corrupted triplet

```
def _corrupt_triple(self, triple, triple_set):
    # Randomly replace either head or tail
    if torch.rand(1) < 0.5:  # Replace head
        while True:
            corrupted_head = torch.randint(self.num_entities, (1,)).item()
            corrupted_triple = (corrupted_head, triple[1], triple[2])
            # If corrupted triple is not in the original set, break the loop
            if corrupted_triple not in triple_set:
                break
        return corrupted_triple
    else:  # Replace tail
        while True:
            corrupted_tail = torch.randint(self.num_entities, (1,)).item()
            corrupted_triple = (triple[0], triple[1], corrupted_tail)
            # If corrupted triple is not in the original set, break the loop
            if corrupted_triple not in triple_set:
                break
        return corrupted_triple
```

Implementation

Score function

```
def score(self, triple):
    head, relation, tail = triple
    if self.norm == 'l1':
        # Compute L1 norm
        return torch.norm(self.entity_embeddings[head] + self.relation_embeddings[relation] - self.entity_embeddings[tail], p=1)
    else:  # self.norm == 'l2'
        # Compute L2 norm
        return torch.norm(self.entity_embeddings[head] + self.relation_embeddings[relation] - self.entity_embeddings[tail], p=2)
```

Train function

```
def train(self, triples, num_epochs, validation_triples=None, batch_size=128):
  epoch_losses = []

  triples = torch.tensor(triples, device=self.device)
  triple_set = set(tuple(t) for t in triples.tolist())

  num_batches = len(triples) // batch_size
  if len(triples) % batch_size != 0:
      num_batches += 1
```

Implementation

Normalizing entity_embeddings at each epoch.

```
for epoch in range(num_epochs):
    self.entity_embeddings = self.entity_embeddings / torch.norm(self.entity_embeddings, p=2, dim=1, keepdim=True)

    indices = torch.randperm(len(triples), device=self.device)
    triples = triples[indices]
```

Calculate gradient if margin+d(h+l,t) - d(h'+l, t) > 0, else, loss = 0

```
corrupted_triple = self._corrupt_triple(triple.tolist(), triple_set)

score = self.score(triple)
corrupted_score = self.score(corrupted_triple)

current_loss = torch.clamp(self.margin + score - corrupted_score, min=0)
```

Implementation

Calculate gradient for each batch

L1-norm

```
if current_loss != 0:
    head, relation, tail = triple
    if self.norm == 'l1':
        positive_diff = (self.entity_embeddings[head] + self.relation_embeddings[relation] - self.entity_embeddings[tail]).sign()
        negative_diff = (self.entity_embeddings[corrupted_triple[0]] + self.relation_embeddings[relation] - self.entity_embeddings[corrupted_triple[2]]).sign()

        grad_head[head] += positive_diff
        grad_tail[tail] -= positive_diff
        grad_relation[relation] += positive_diff - negative_diff
        grad_head[corrupted_triple[0]] -= negative_diff
        grad_head[corrupted_triple[2]] += negative_diff
```

L2-norm

```
else:
    positive_diff = 2 * (self.entity_embeddings[head] + self.relation_embeddings[relation] - self.entity_embeddings[tail])
    negative_diff = 2 * (self.entity_embeddings[corrupted_triple[0]] + self.relation_embeddings[relation] - self.entity_embeddings[corrupted_triple[2]])

    grad_head[head] += positive_diff
    grad_tail[tail] -= positive_diff
    grad_relation[relation] += positive_diff - negative_diff
    grad_head[corrupted_triple[0]] -= negative_diff
    grad_head[corrupted_triple[2]] += negative_diff
```

Implementation

Update embeddings for each batch

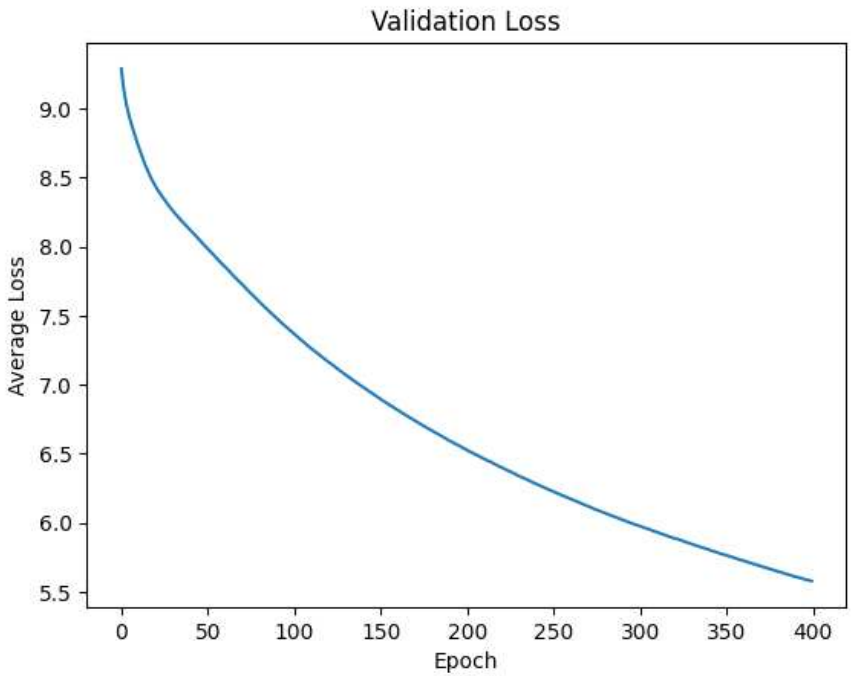
```
self.entity_embeddings -= self.learning_rate * grad_head
self.entity_embeddings -= self.learning_rate * grad_tail
self.relation_embeddings -= self.learning_rate * grad_relation
```

FB15k

```
transe = TransE(num_entities=num_entities, num_relations=num_relations, embedding_dim=50, learning_rate=0.01, margin=1.0, norm='l1')
print(transe.device)
transe.train(train_data, validation_triples=valid_data, num_epochs=50, batch_size=128)
```

Implementation

FB15K



Validation Loss :L1-norm of validation set

k = 50, learning rate=0.01, margin=1.0, norm=L1

Entities	Relationships	Train	Validation	Test
14951	1345	483142	50000	59071