UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

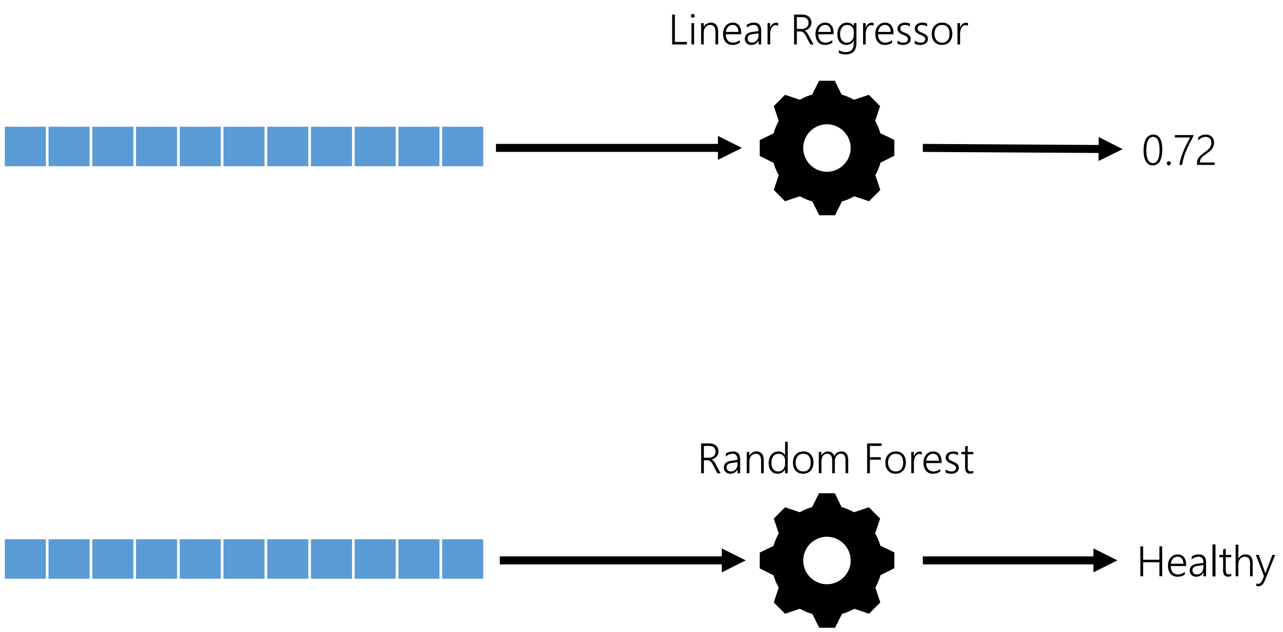
Knowledge Graph Embeddings

SDM LAB-2

14.5.2025

Context: Embeddings

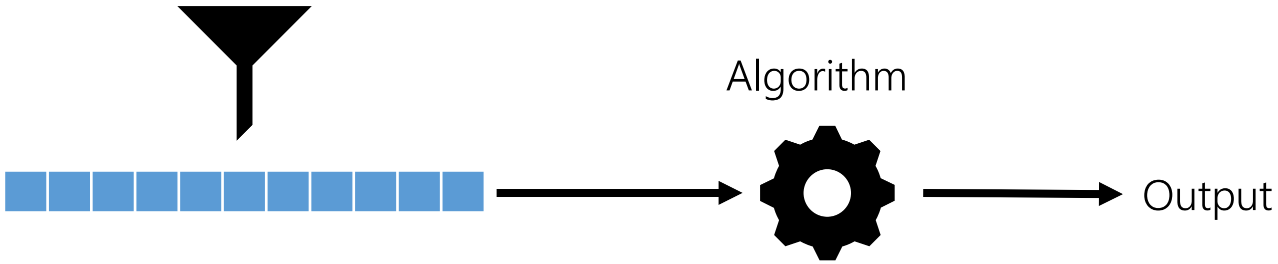
Usually, Machine Learning algorithms expect **vectors** as the input of their models:



Context: Embeddings

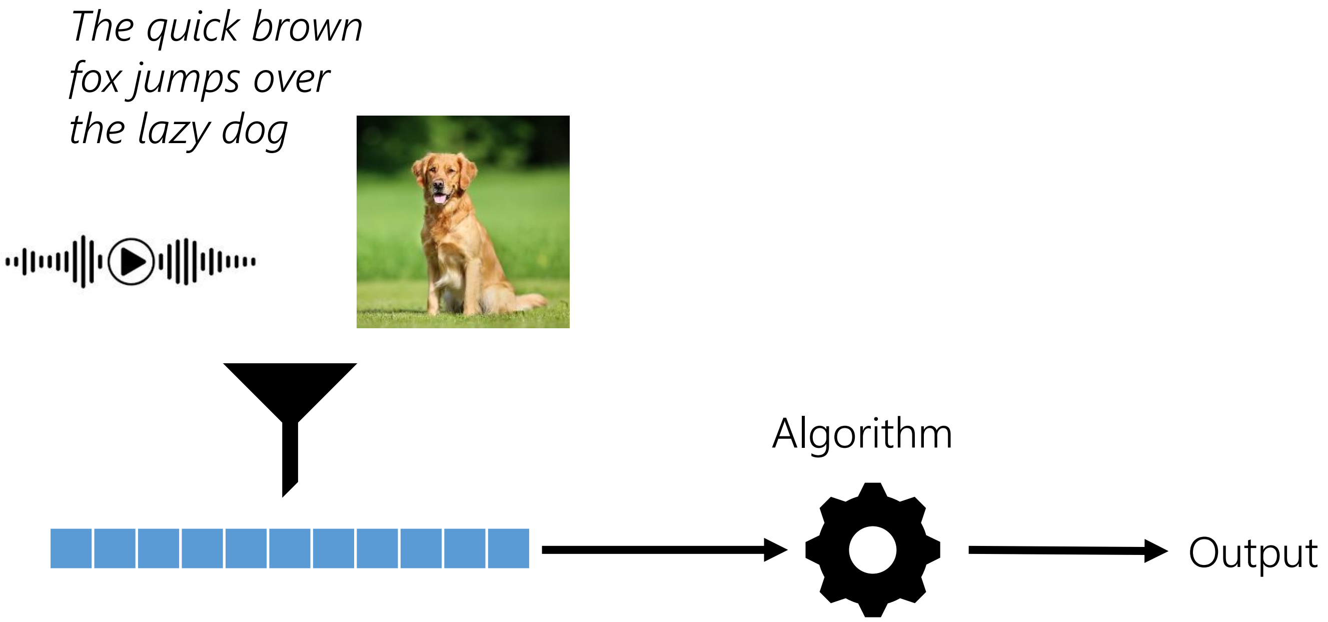
These transformations from data to vectors are straightforward when the input is **tabular**, but can be problematic for **other data representations**.

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
98	0,4	High	1	56,4	1,32
76	0,7	Low	5	43,7	1,11
45	0,8	Low	7	94,2	0,78
...	...	...	...	...	...



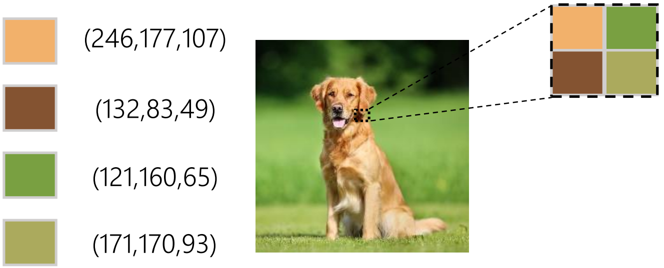
Context: Embeddings

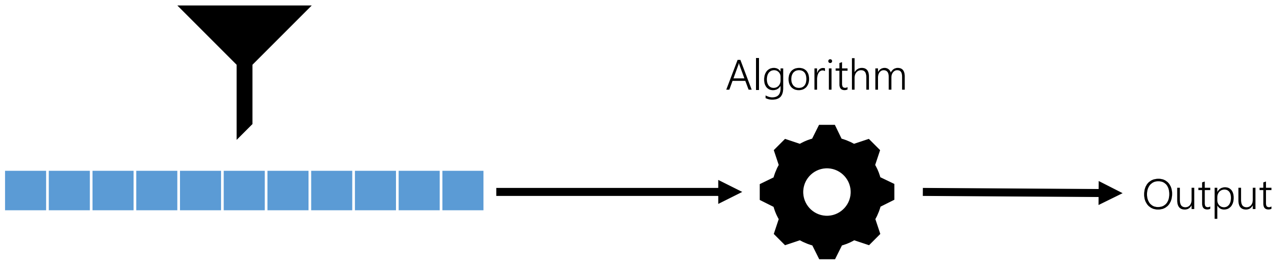
These transformations from data to vectors are straightforward when the input is **tabular**, but can be problematic for **other data representations**.



Context: Embeddings

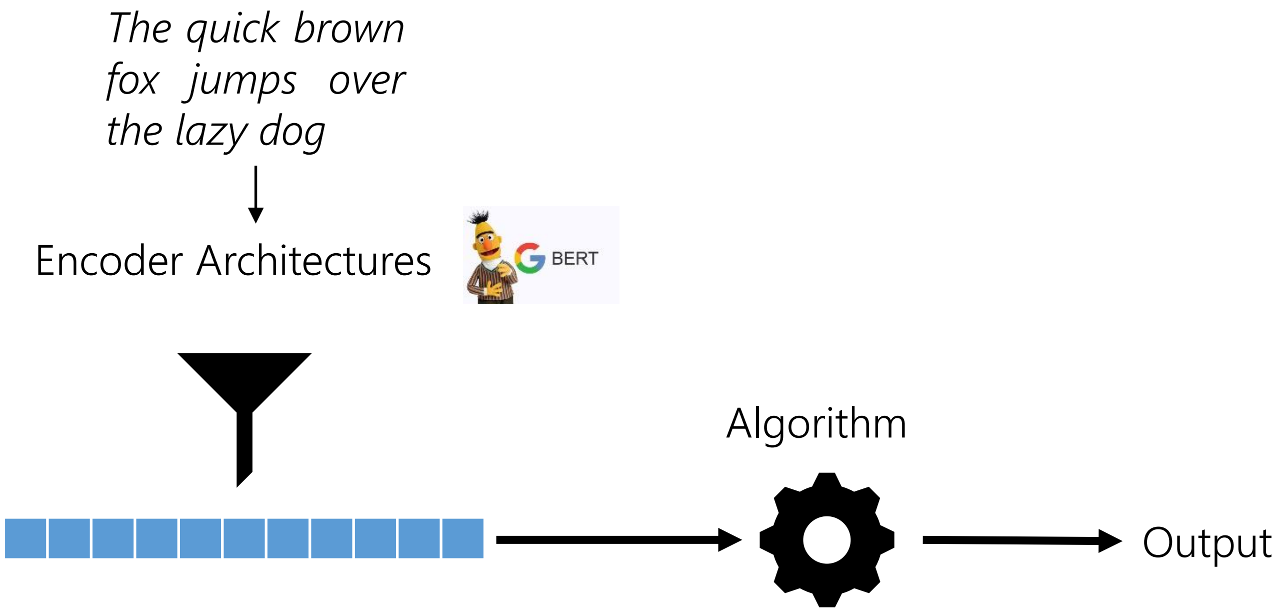
These transformations from data to vectors are straightforward when the input is **tabular**, but can be problematic for **other data representations**.





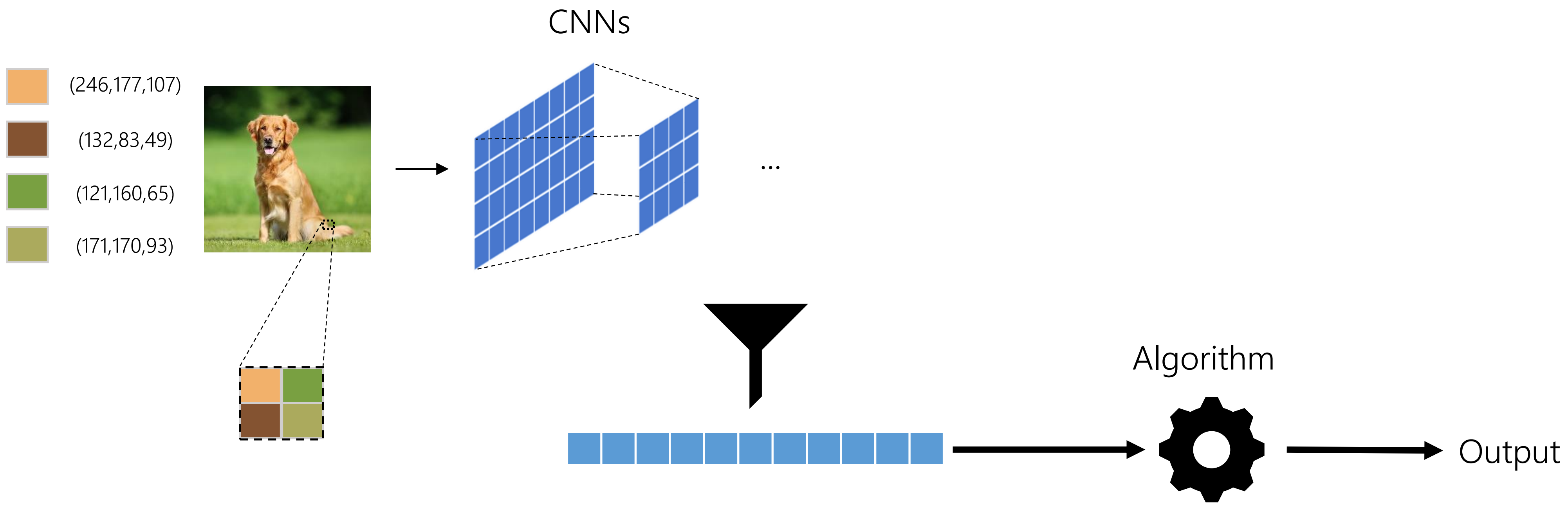
Context: Embeddings

These transformations from data to vectors are straightforward when the input is **tabular**, but can be problematic for **other data representations**.



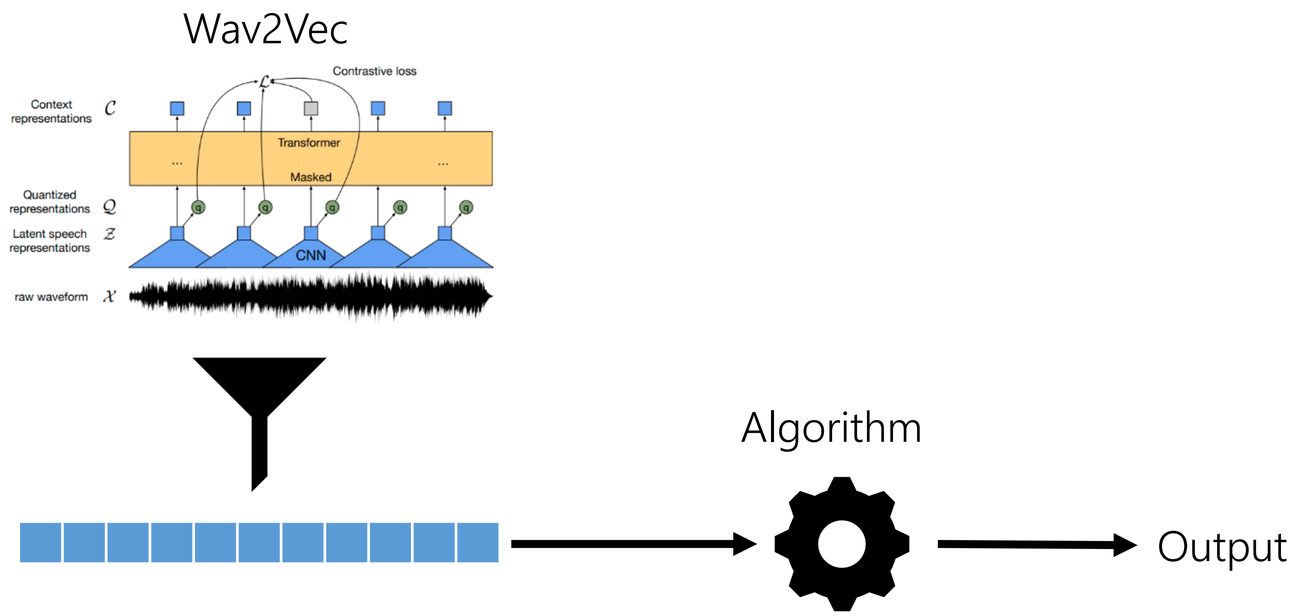
Context: Embeddings

These transformations from data to vectors are straightforward when the input is **tabular**, but can be problematic for **other data representations**.



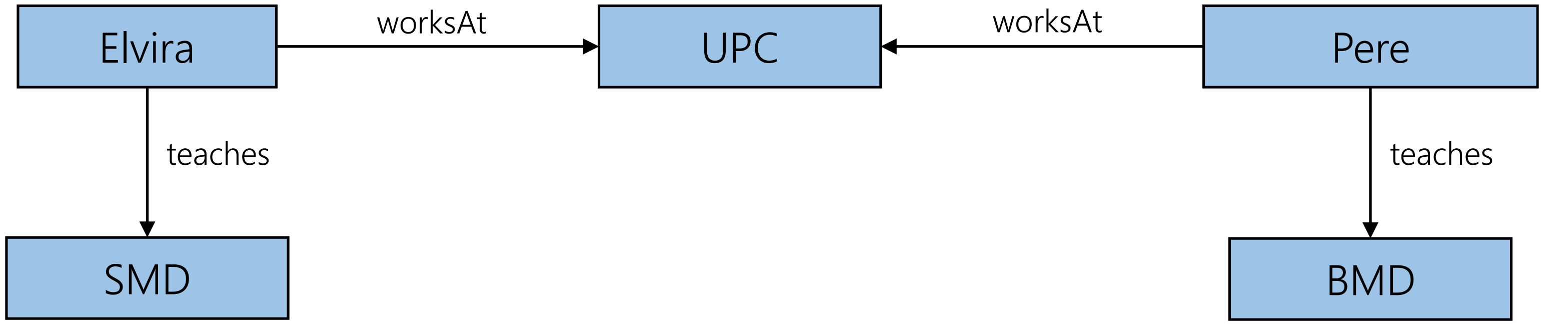
Context: Embeddings

These transformations from data to vectors are straightforward when the input is **tabular**, but can be problematic for **other data representations**.



Knowledge Graph Embeddings

Knowledge Graphs are a network of **entities** interconnected by **relations** between them, giving semantics to these links, making it both *machine-readable* and *human-interpretable*.

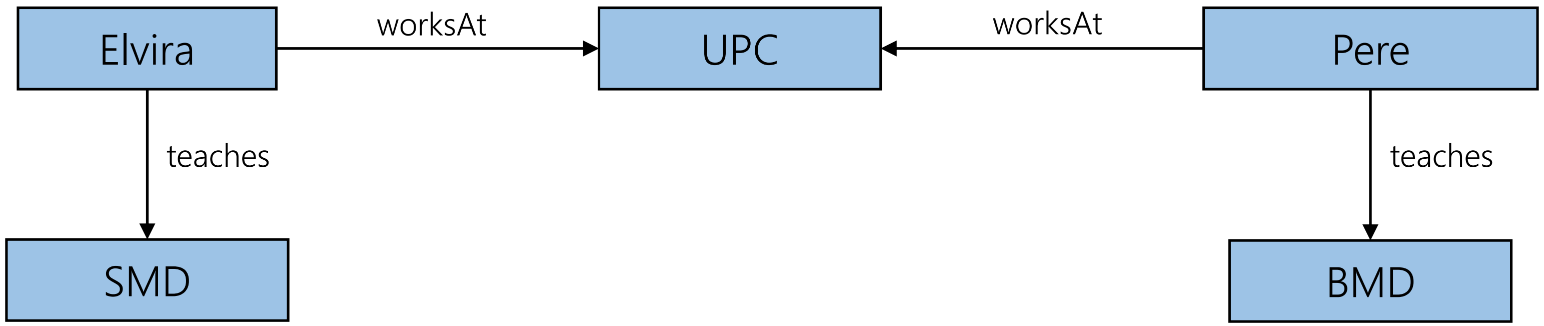


They can be represented using **triples:**

$$(\underbrace{\textit{Pere}}_{\textit{head entity}},\ \underbrace{\textit{worksAt}}_{\textit{relation}},\ \underbrace{\textit{UPC}}_{\textit{tail entity}})$$

Knowledge Graph Embeddings

Knowledge Graphs already allow to apply some **graph-specific** algorithms, but they are limited and usually computationally expensive.



Path-finding

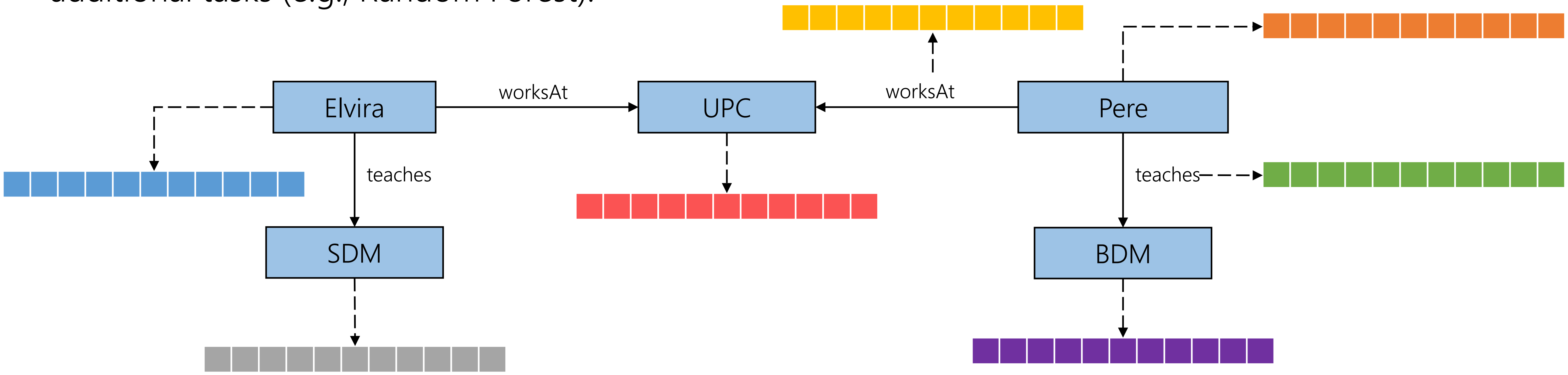
Connectivity

Coloring

…

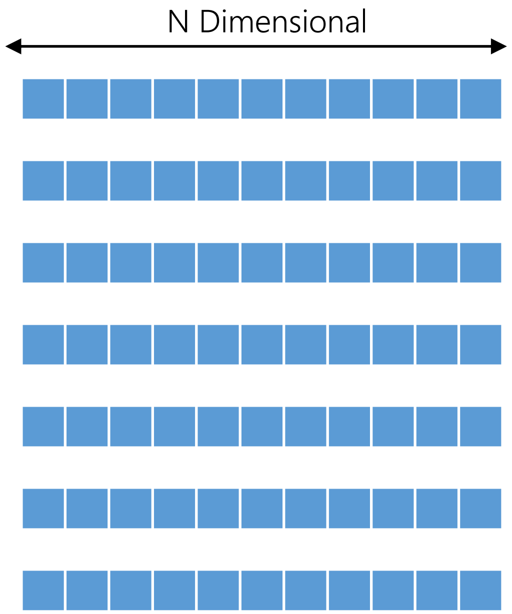
Knowledge Graph Embeddings

The data stored in a KG sometimes also needs to be **transformed into vectors** to serve as the input for additional tasks (e.g., Random Forest).



Knowledge Graph Embeddings

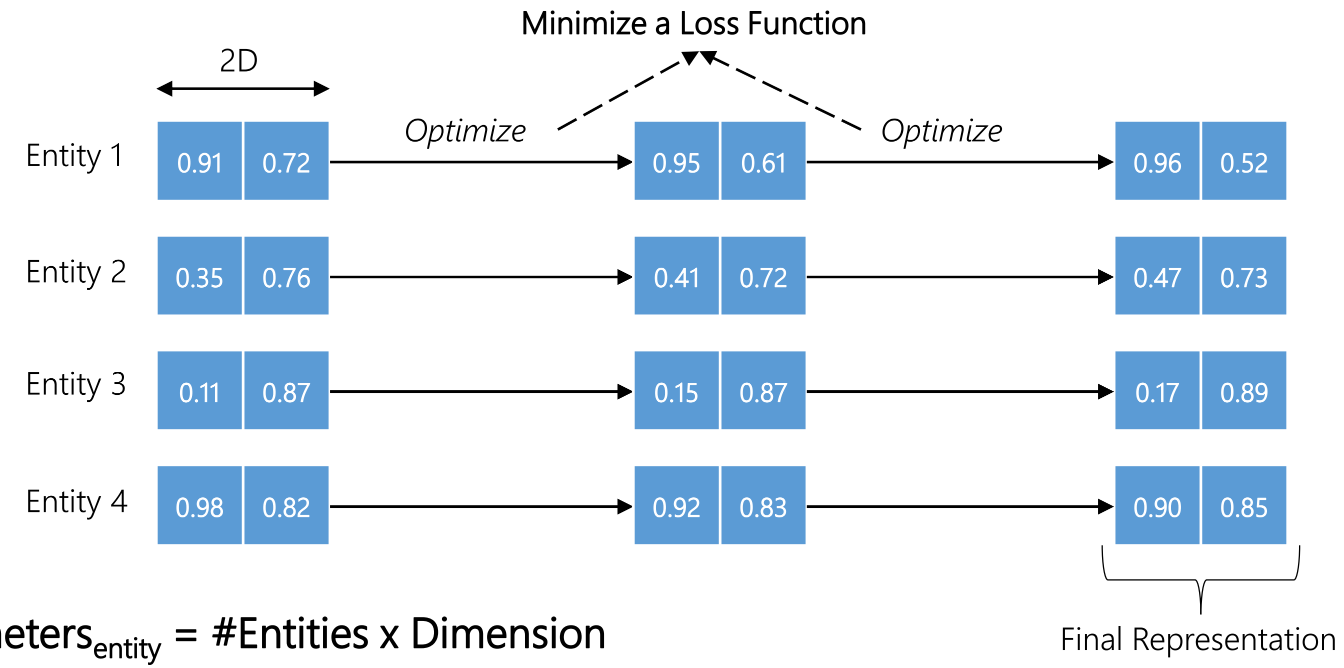
The data stored in a KG sometimes also needs to be **transformed into vectors** to serve as the input for additional tasks (e.g., Random Forest).



The resulting vectors (i.e., Knowledge Graph Embeddings) should **retain** as much contextual/structural/semantic information as possible, so different **KGE Models** have been designed for these purposes.

Knowledge Graph Embeddings

In the KGE Models, the embeddings are treated as **parameters,** which are optimized based on some **transformations**, defining the KGE Model.

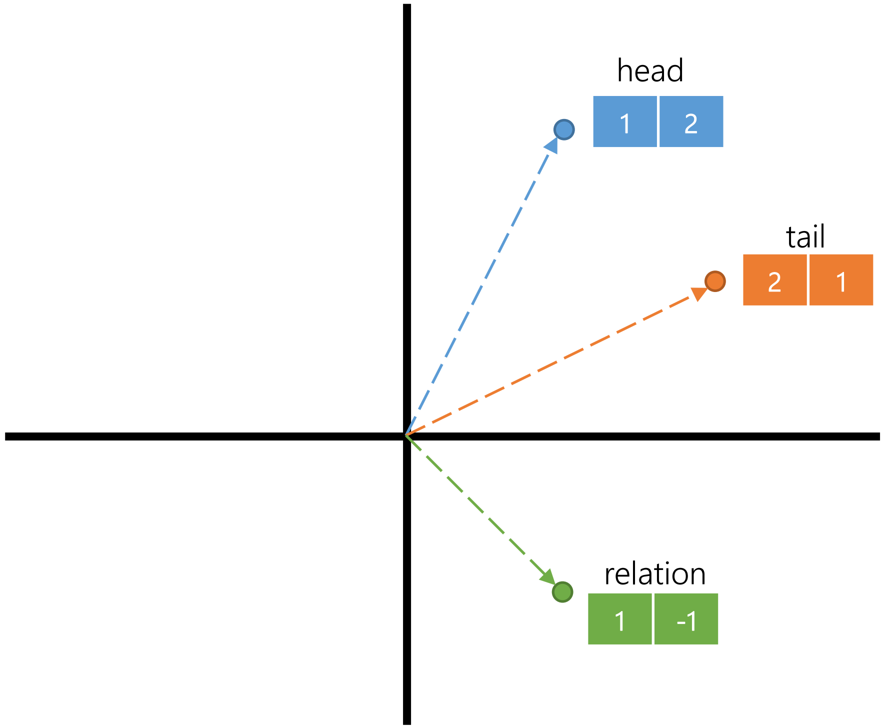


$\text{Parameters}_{\text{entity}}$ = #Entities x Dimension

KGE Models: Translational

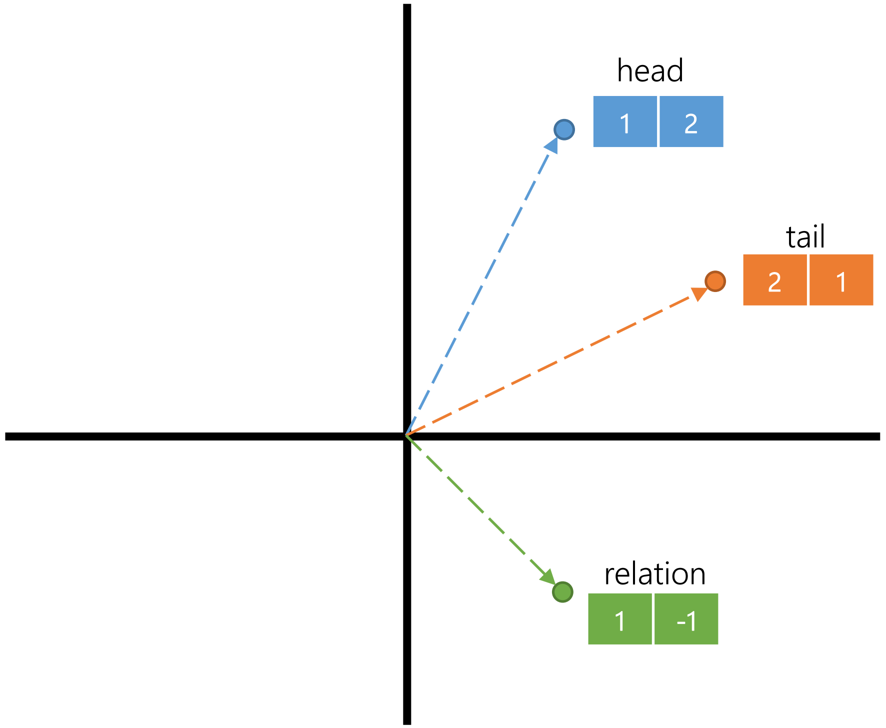
Translational models are the most simple ones, and are based on geometric transformations of the embeddings.

(head, relation, tail)



KGE Models: Translational

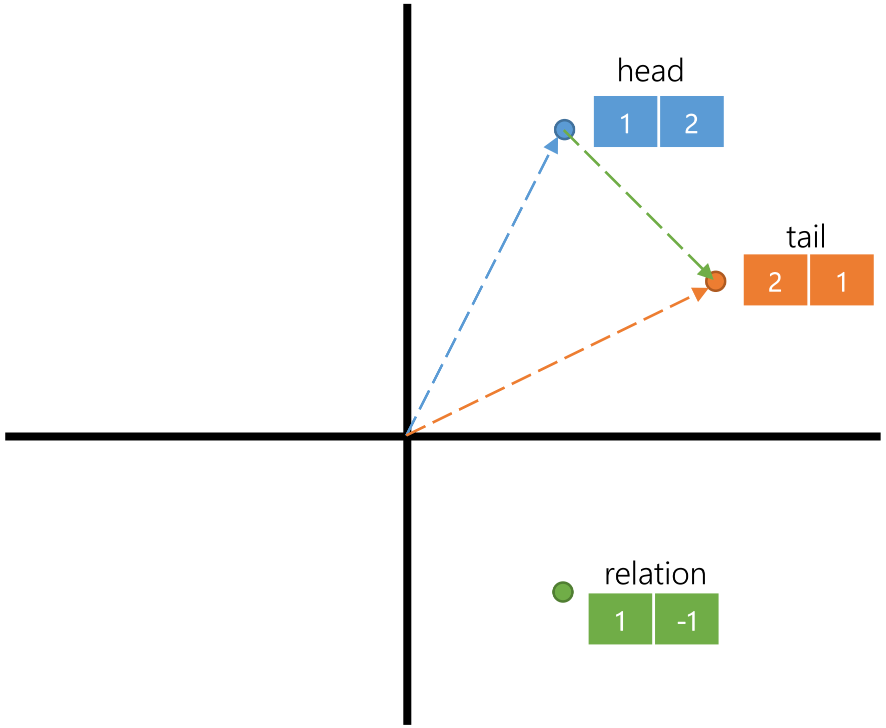
Example of a Loss Function: <u>TransE</u>: Minimize the translational distance from the *head entity* to the *tail entity* by using the *relation embedding*.



TransE: Loss= |head + relation - tail|

KGE Models: Translational

Example of a Loss Function: <u>TransE</u>: Minimize the translational distance from the *head entity* to the *tail entity* by using the *relation embedding*.



TransE: Loss= |head + relation - tail|

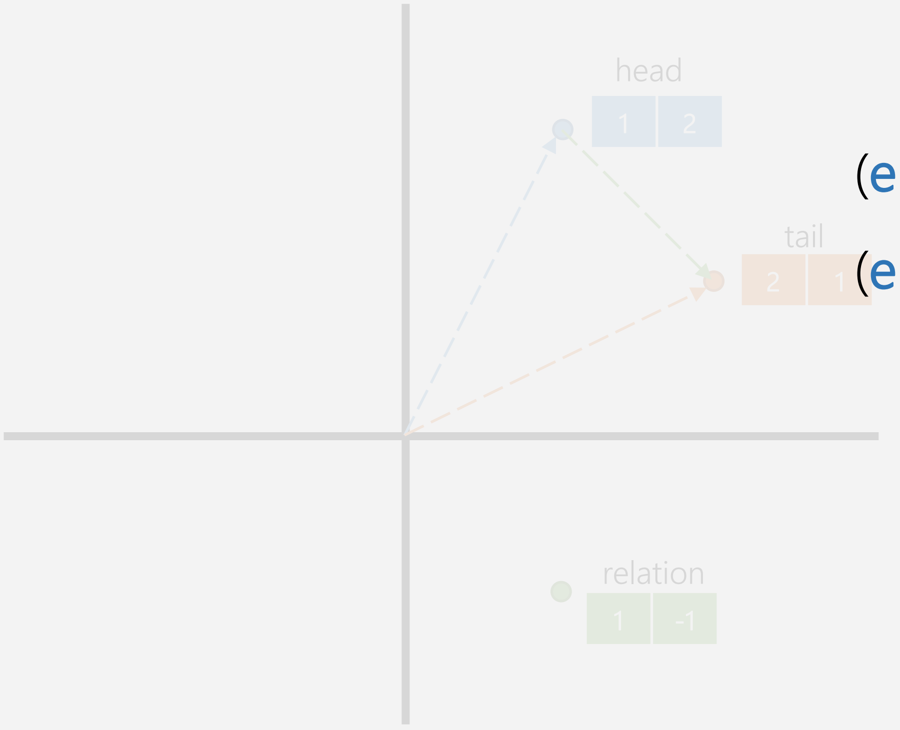
(head, relation, tail)

Loss= |(1,2)+(1,-1) – (2,1)| = 0

KGE Models: Translational

LIMITATIONS

Example of a Loss Function: <u>TransE</u>: Minimize the translational distance from the *head entity* to the *tail entity* by using the *relation embedding*.



Symmetry

TransE: Loss= |head + relation - tail|

(entity1, relation, entity2)

(entity2, relation, entity1)

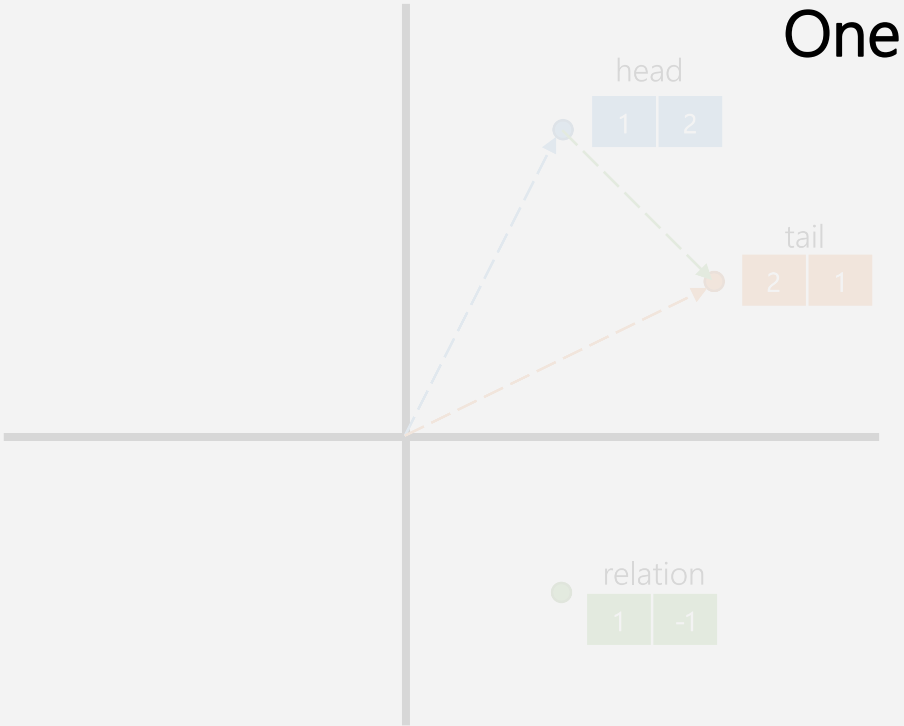
(head, relation, tail)

$Loss = |(1,2)+(1,-1) - (2,1)| = 0$

KGE Models: Translational

LIMITATIONS

Example of a Loss Function: TransE: Minimize the translational distance from the *head entity* to the *tail entity* by using the *relation embedding*.



One-to-Many | Many-to-one

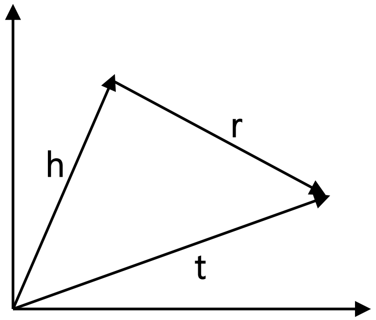
TransE: Loss= |head + relation - tail|

(entity1, relation, entity2)

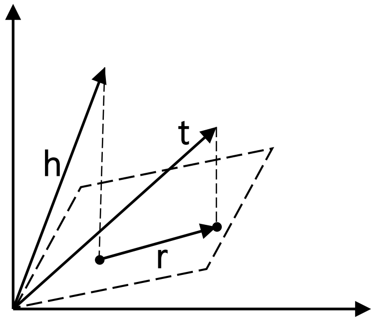
(entity1, relation, entity3) (head, relation, tail)

$$Loss= |(1,2)+(1,-1) - (2,1)| = 0$$

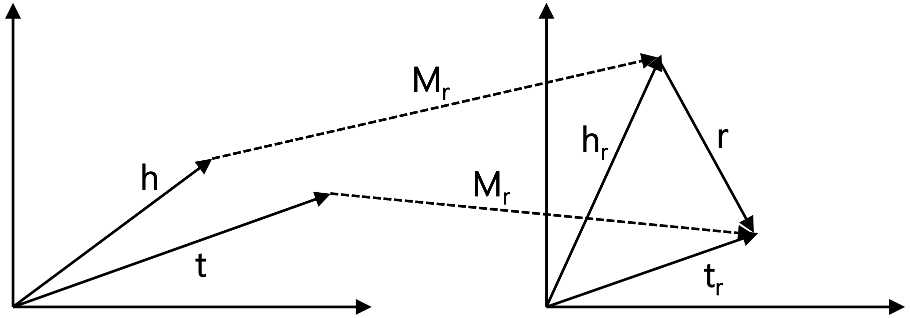
KGE Models: Translational



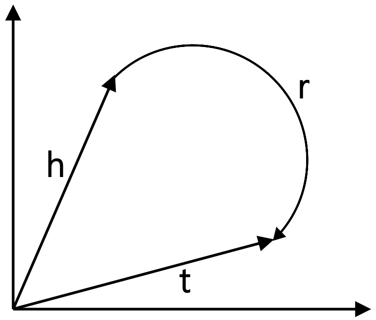
TransE



TransH



TransR



RotatE

KGE Models: Semantic

Semantic models try to capture **latent semantics** of entities and relations, by maximizing the plausibility of a triple.

Rescal: $head^T * M_{relation} * tail$

DistMult: $head^T * relation * tail$

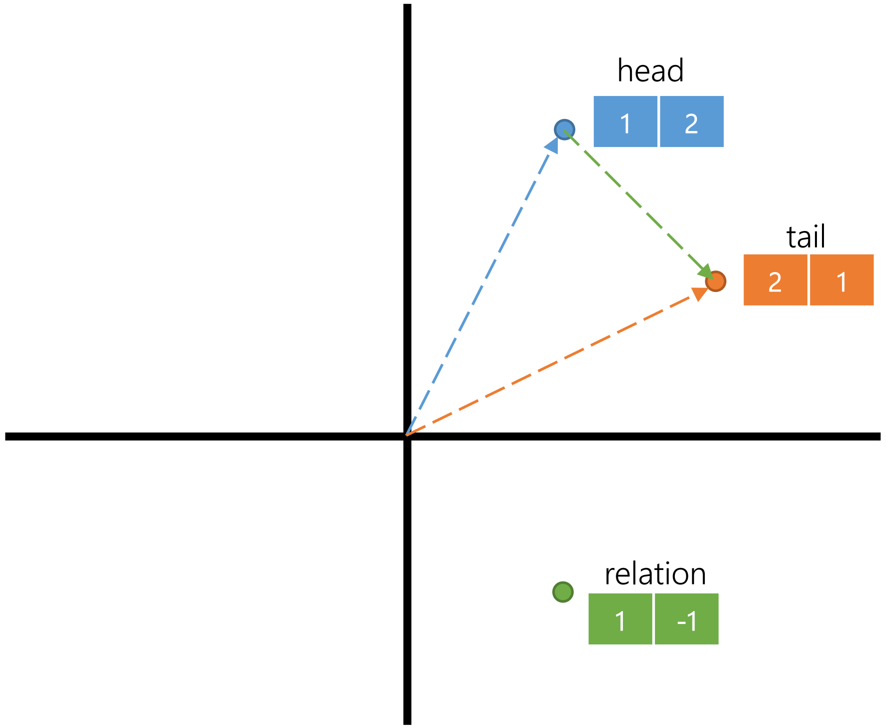
HolE: $relation^T * (head \circ tail)$

KGE Models: Neural Netwok

Maximize the plausibility of a triple, including **Neural Network functions** to learn hidden relationships.

ConvE, ConvKB: 1 or 2D convolutions, ReLU activation functions,…

KGE Models: Learning

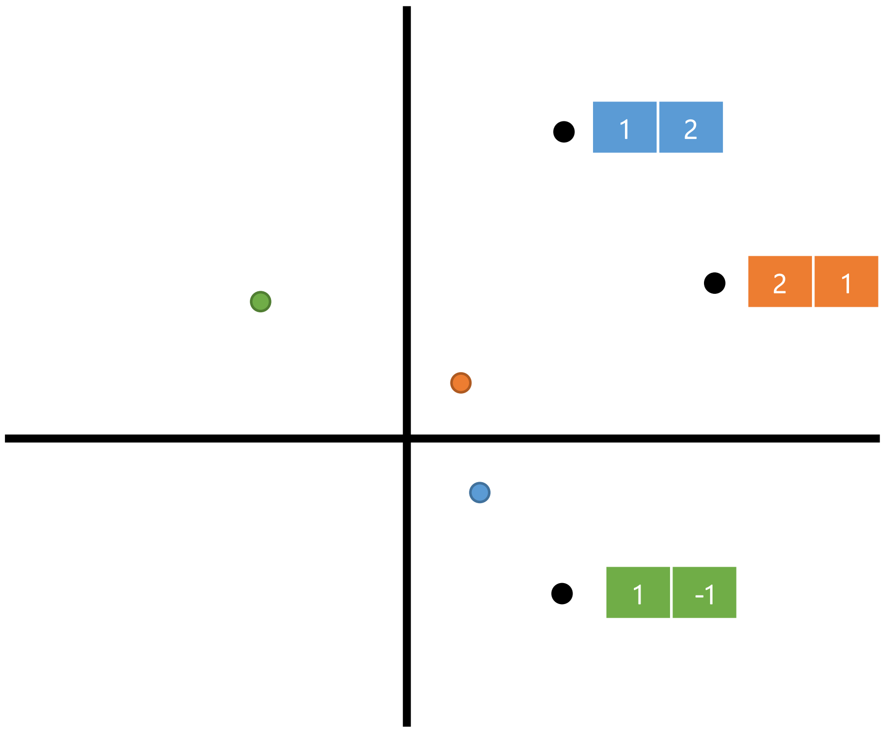


TransE: Loss= |head + relation - tail|

$$Loss = |(1,2) + (1,-1) - (2,1)| = 0$$

KGE Models: Learning

Step 0



TransE: Loss= |head + relation - tail|

$$Loss = |(0.4,-0.4)+(-1,0.9) - (0.3,0.3)| = 0.92$$

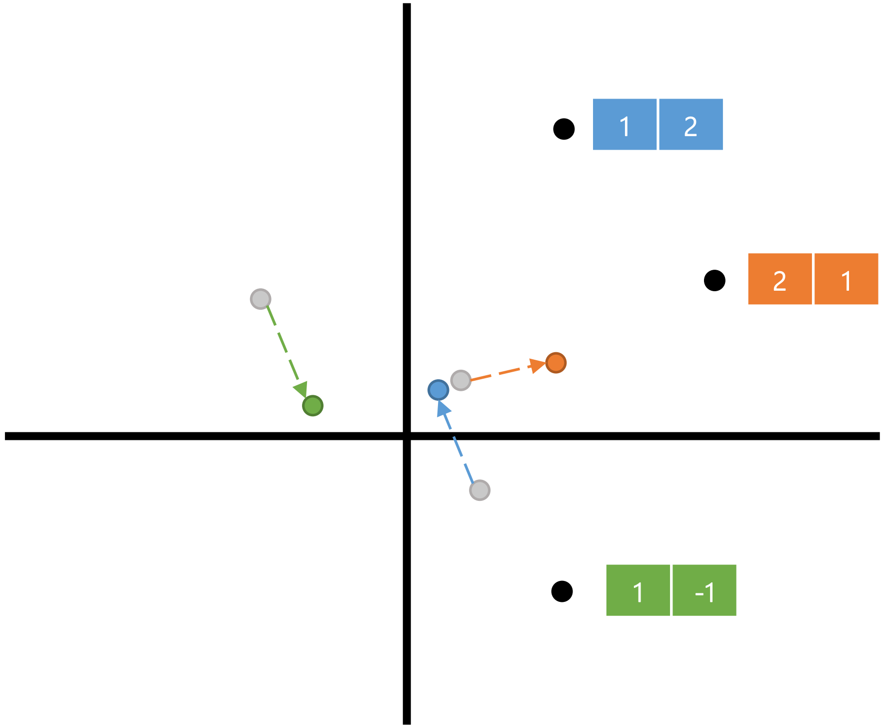
Changes parameters that affect directly or indirectly the loss function

● Optimal Position

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

KGE Models: Learning

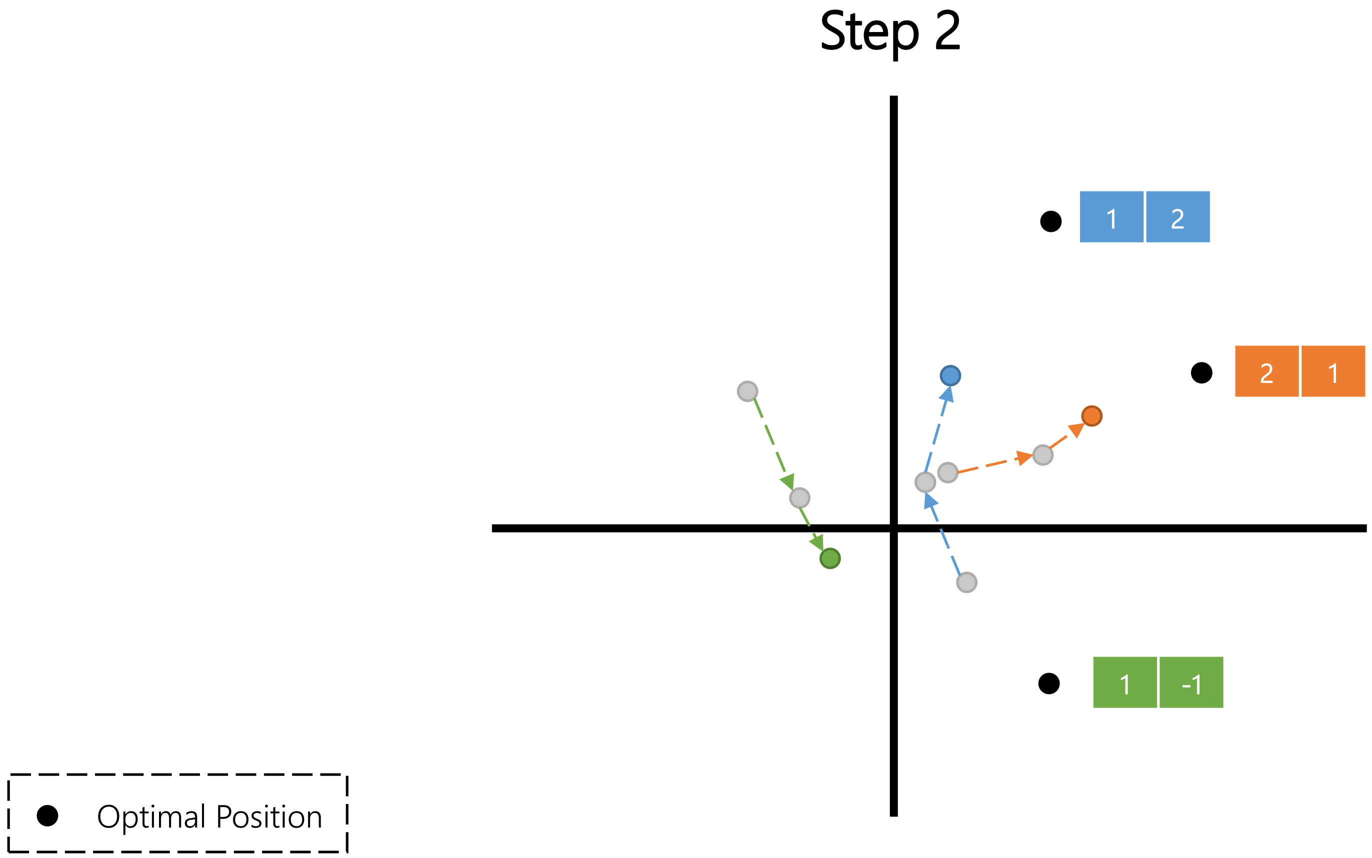
Step 1



TransE: Loss= |head + relation - tail|

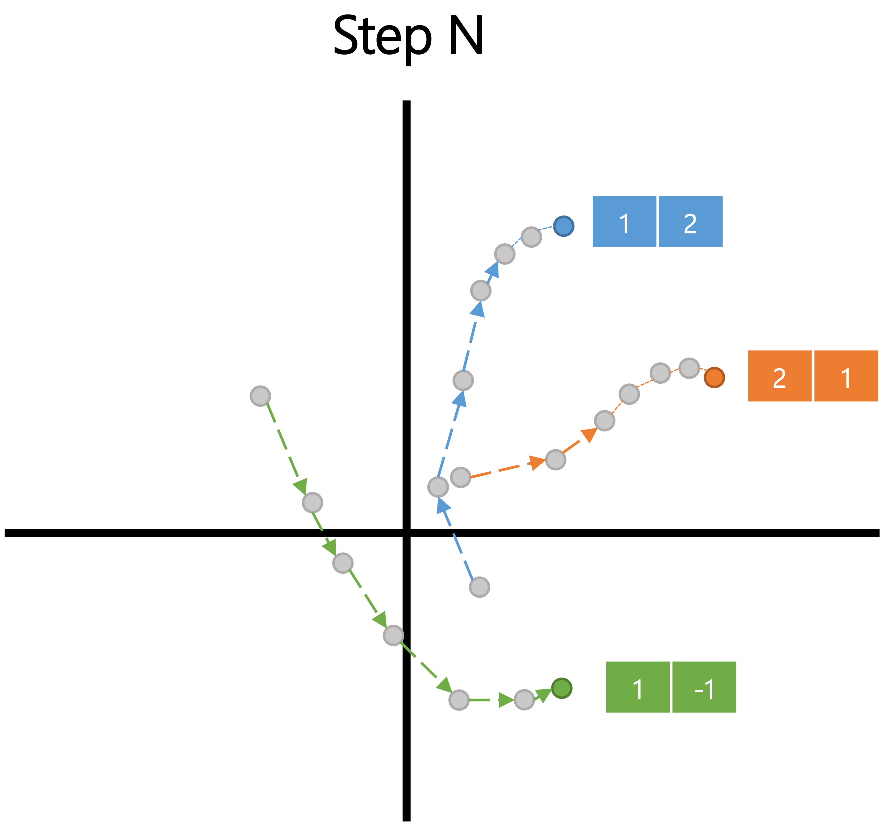
$$Loss = |(0.2,-0.2)+(-0.5,0.1) - (0.8,0.4)| = 0.70$$

KGE Models: Learning



TransE: Loss= |head + relation - tail|

KGE Models: Learning

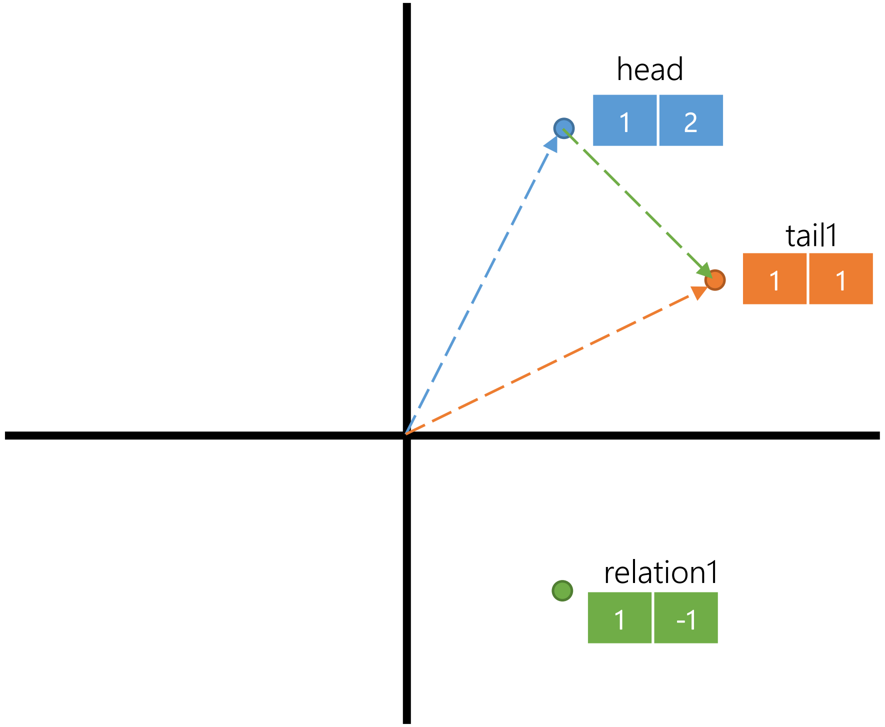


TransE: Loss= |head + relation - tail|

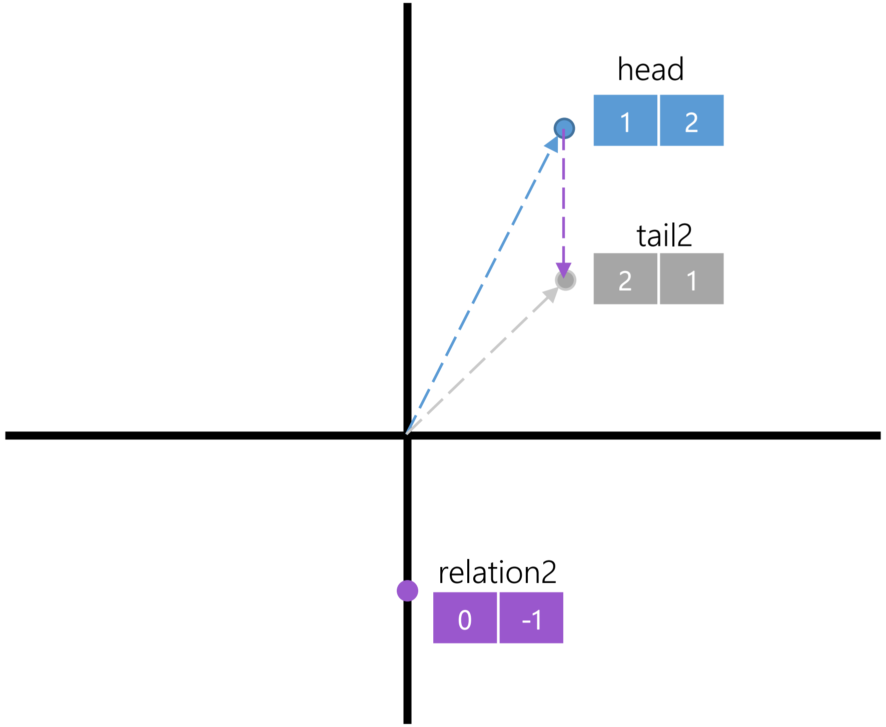
$$Loss = |(1,2) + (1,-1) - (2,1)| = 0$$

● Optimal Position

KGE Models: Learning

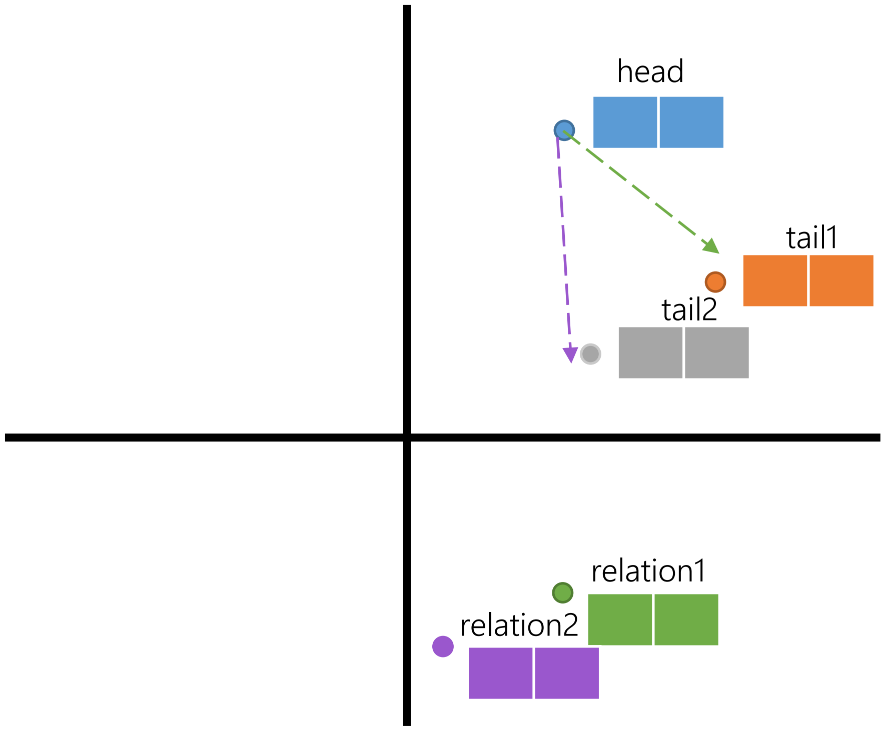


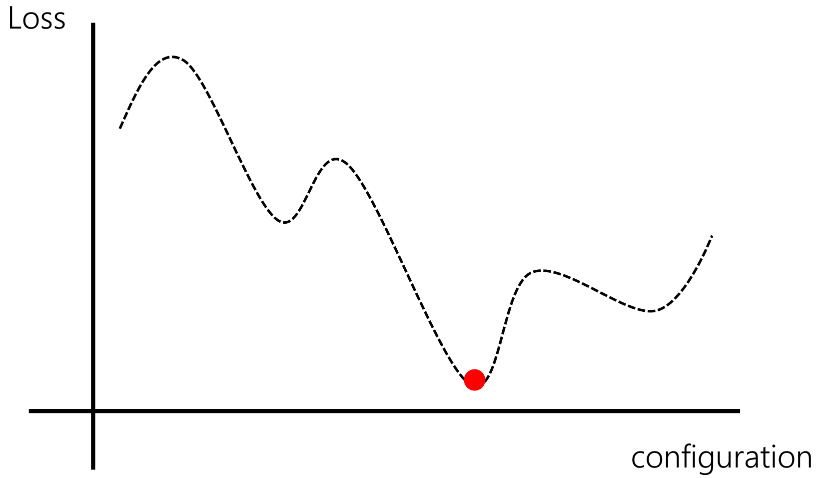
(head, relation1, tail1)



(head, relation2, tail2)

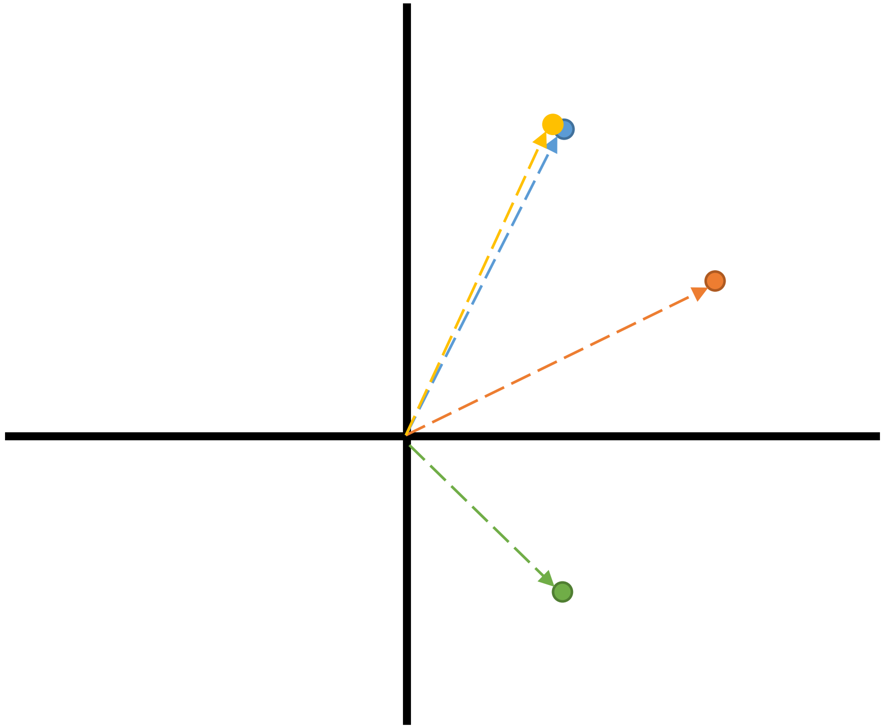
KGE Models: Learning





(head, relation1, tail1)
(head, relation2, tail2)

KGE Models: Learning



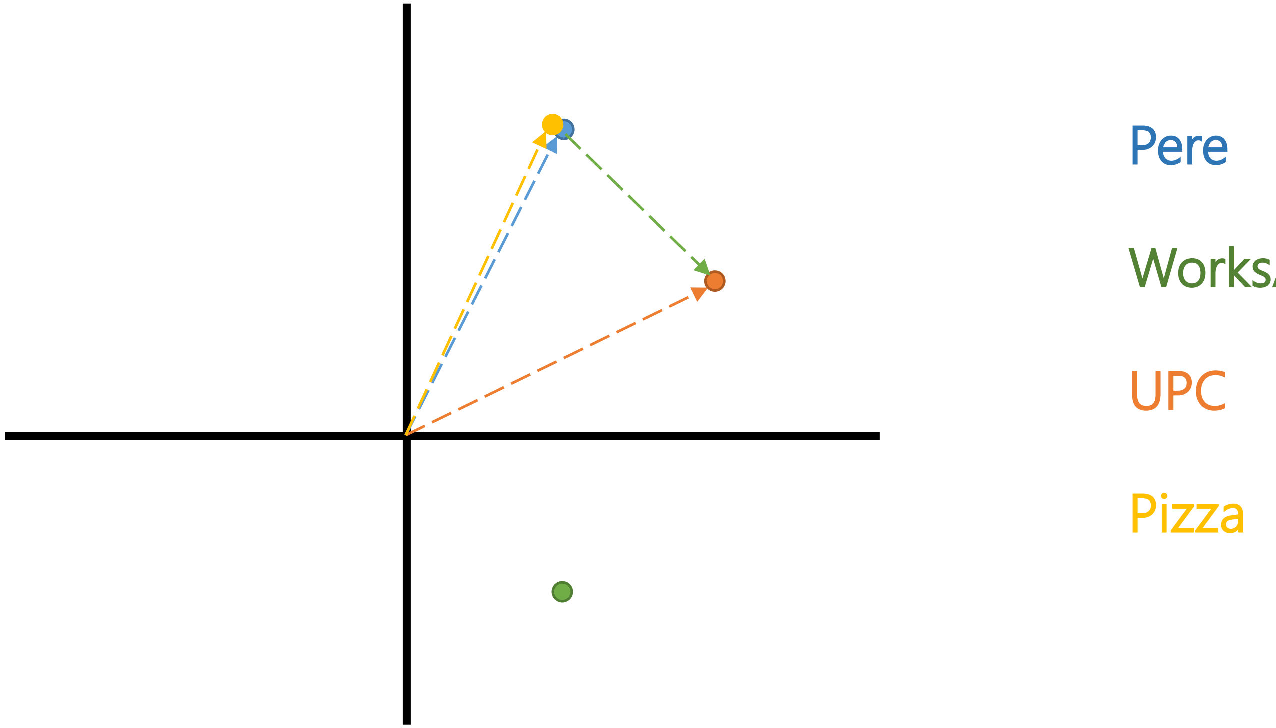
Pere

WorksAt

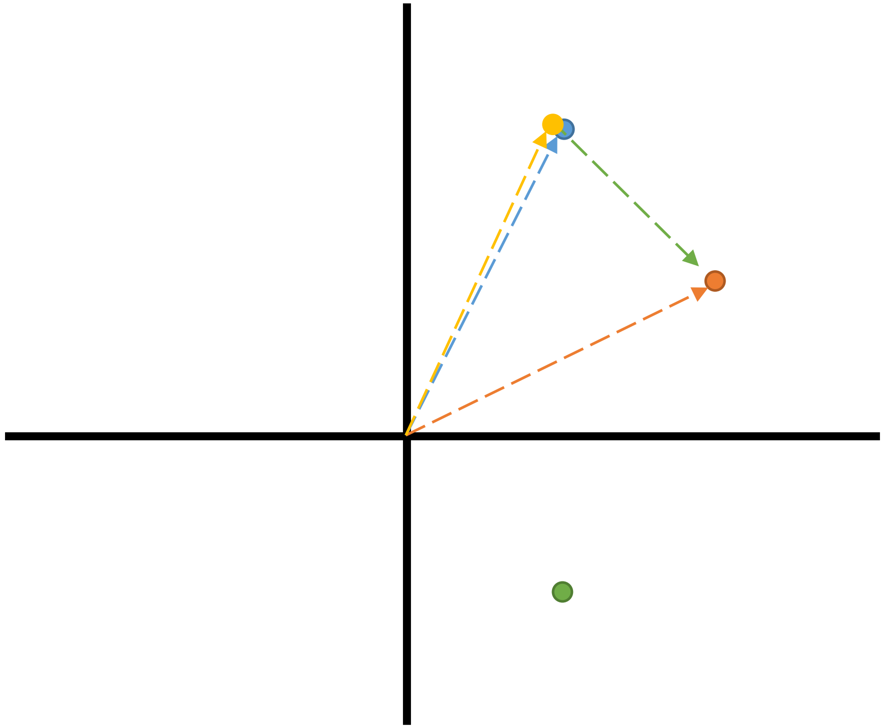
UPC

Pizza

KGE Models: Learning



KGE Models: Learning



Pere

WorksAt

UPC

Pizza

KGE Models: Learning - Margin Loss

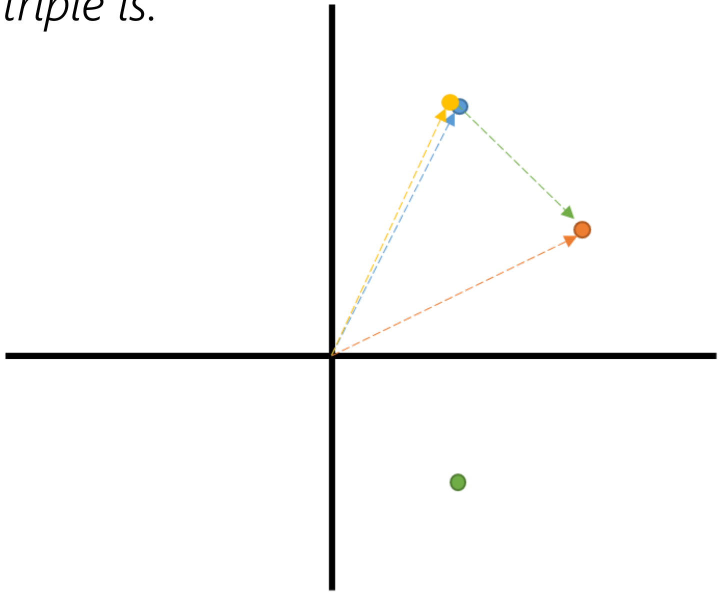
To solve this issues, **negative triples** are generated and the loss structure is changed:

Loss = max{-Score(Positive Triple) + Score(Negative Triple) + margin, 0}

The higher the score, the more probable a triple is.

hyperparameter

Different generation mechanisms



(head, relation, tail)

(head, relation, tail) (head, relation, tail)

KGE Models: Learning - Parameters

- Model
- Margin
- Number of Negatives samples per Positive Sample
- Embedding Dimension
- Learning Rate
- Epochs

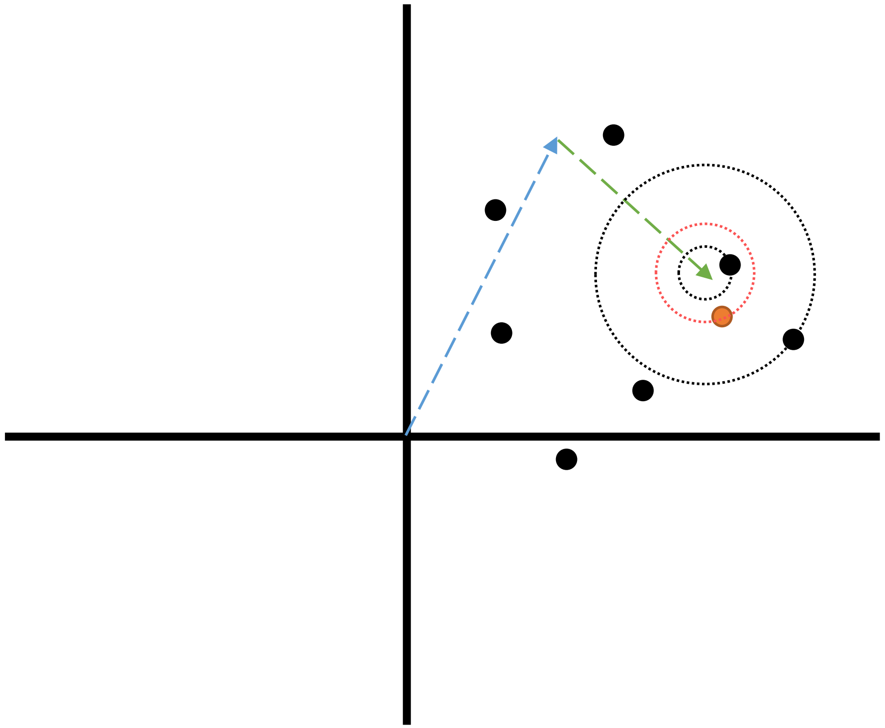
KGE Models: Evaluation

The standard way of evaluation Knowledge Graph Embeddings is by solving the **Link Prediction** task:

From the test dataset, one of the entities of the triple is removed (e.g., [head, relation, ?]), and we try to predict which was the original entity.

KGE Models: Evaluation

Hits@3



TEST SET

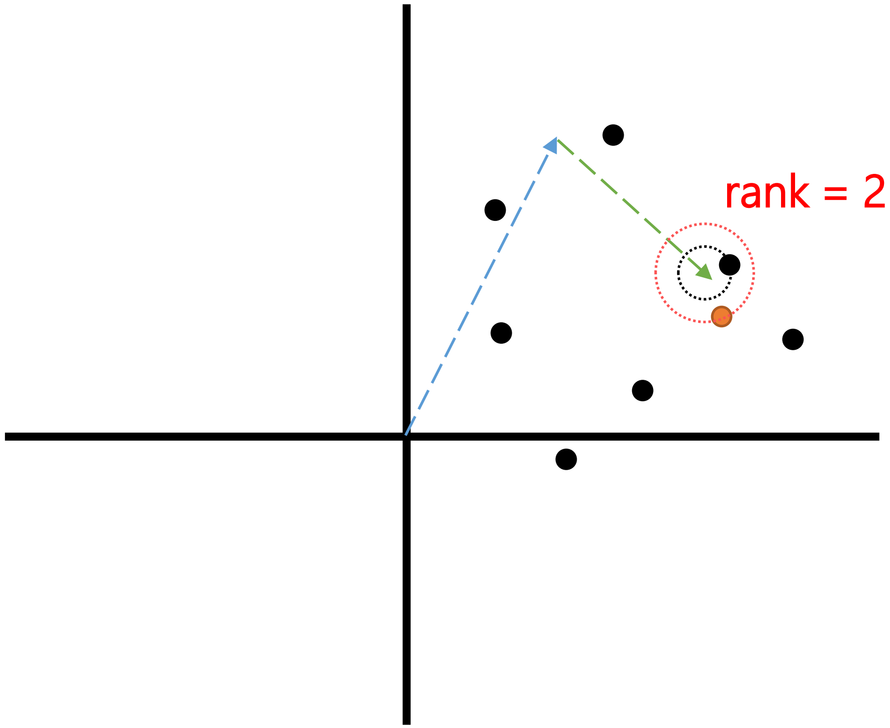
(head,relation,tail) ⟶ (head,relation,?)

$$tail \approx head + relation$$

Hits@3: Proportion of test triples in which the correct missing entity is the first, second or third "closest" entity.

KGE Models: Evaluation

Mean Reciprocal Rank (MRR)



TEST SET

(head,relation,tail) ⟶ (head,relation,?)

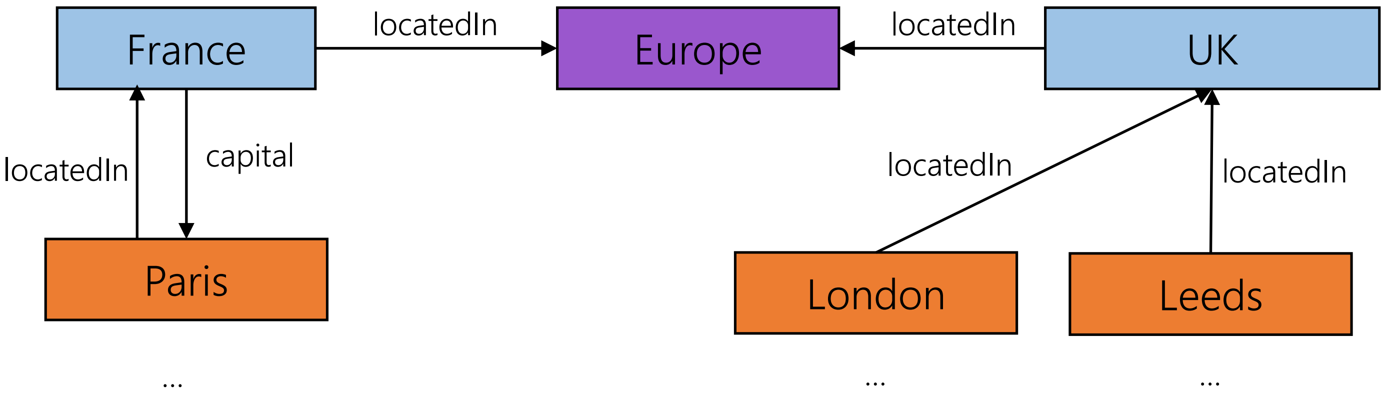
$$tail \approx head + relation$$

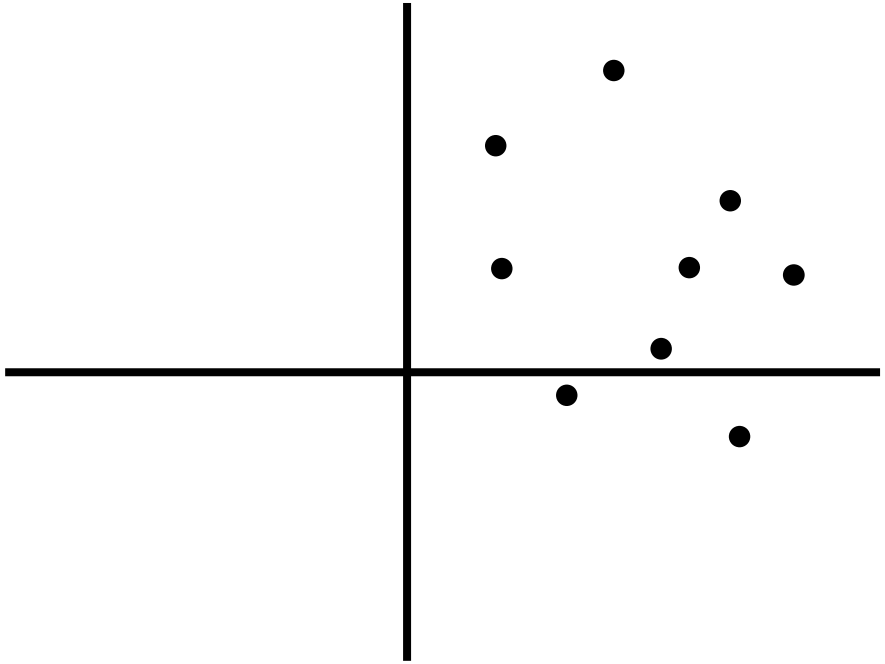
$$MRR = \frac{1}{|N|}\sum_{i=1}^{N}\frac{1}{rank_i}$$

KGE Models: Applications

Generating KGE allows to use well-know **machine learning** models over KG (e.g., classification, clustering,…). However, we can also use them directly:

Knowledge Graph Completion

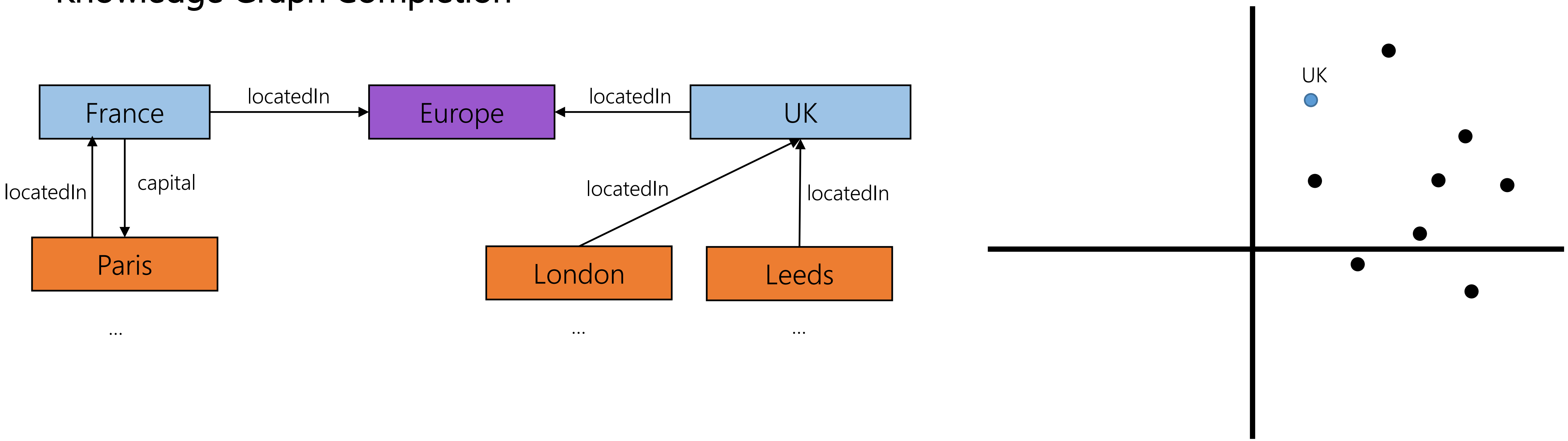




KGE Models: Applications

Generating KGE allows to use well-know **machine learning** models over KG (e.g., classification, clustering,...). However, we can also use them directly:

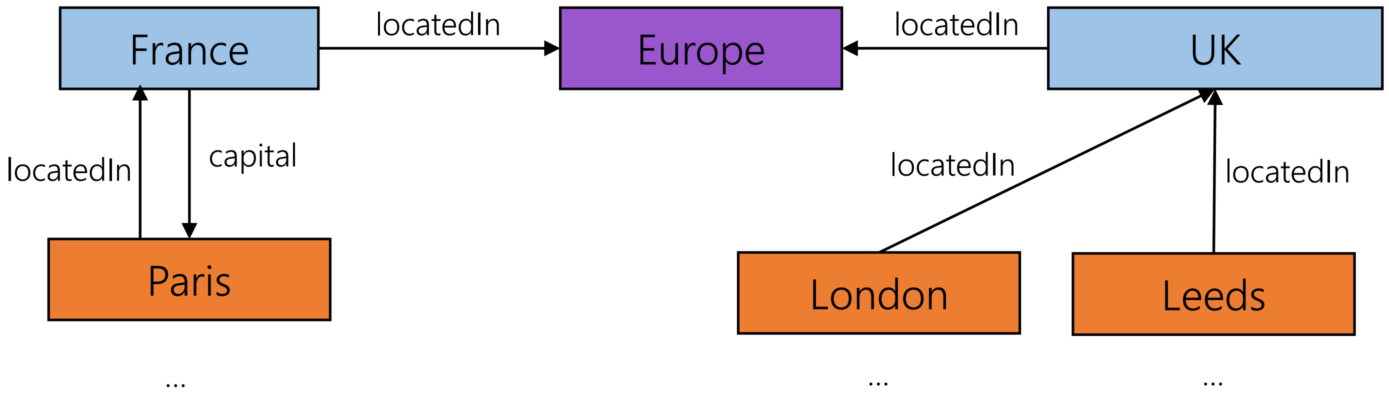
Knowledge Graph Completion

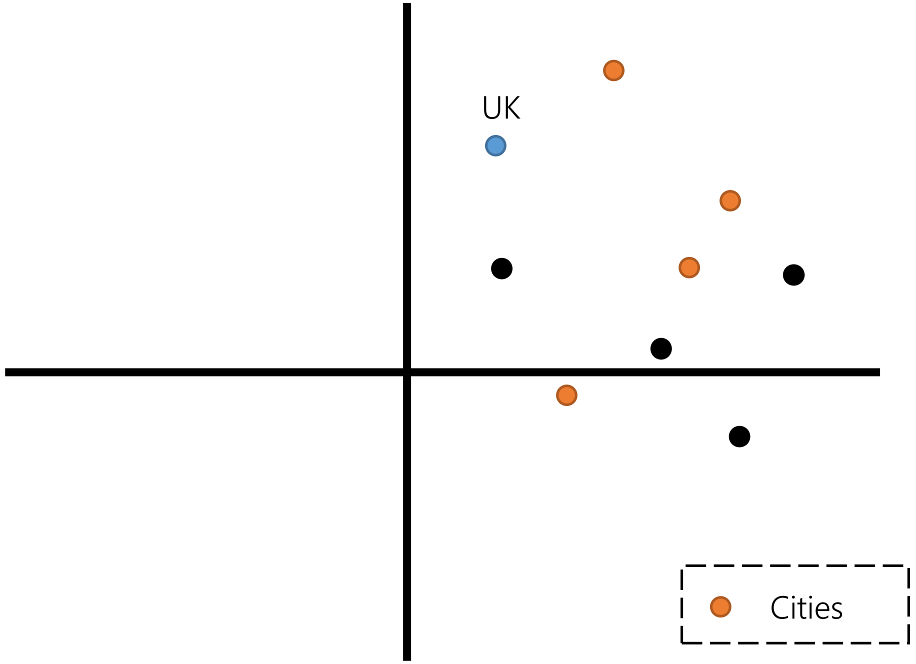


KGE Models: Applications

Generating KGE allows to use well-know **machine learning** models over KG (e.g., classification, clustering,…). However, we can also use them directly:

Knowledge Graph Completion

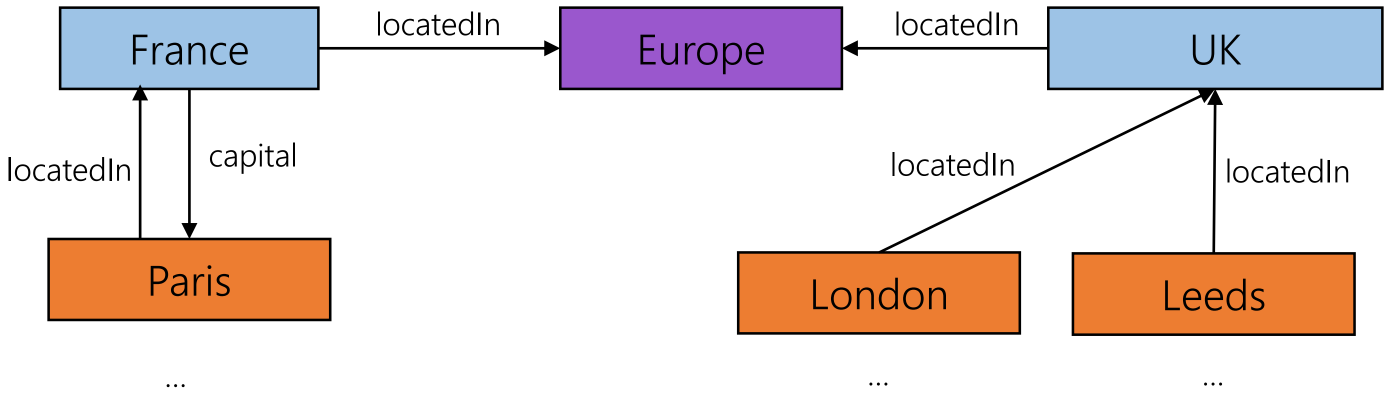


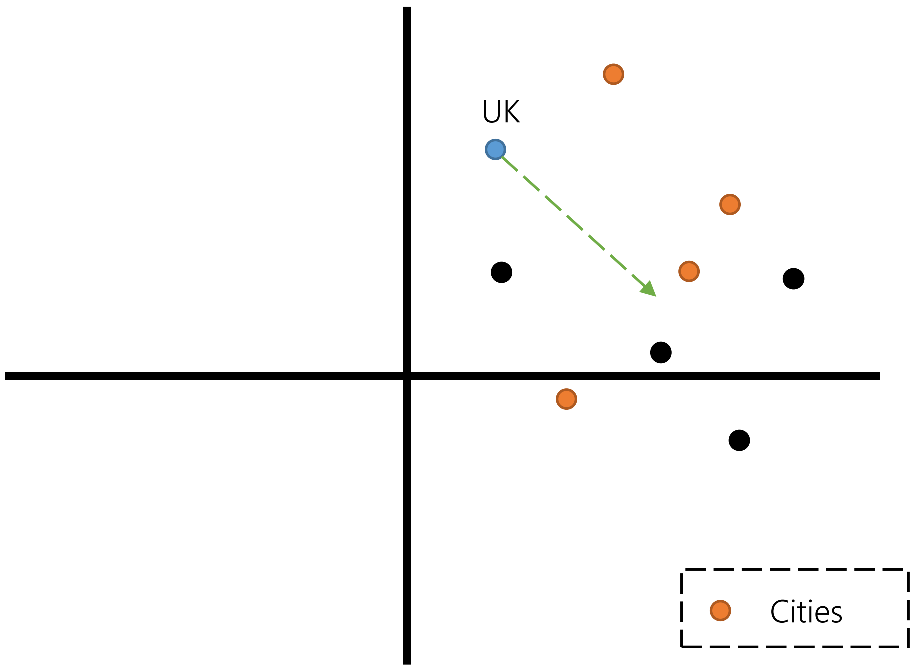


KGE Models: Applications

Generating KGE allows to use well-know **machine learning** models over KG (e.g., classification, clustering,…). However, we can also use them directly:

Knowledge Graph Completion

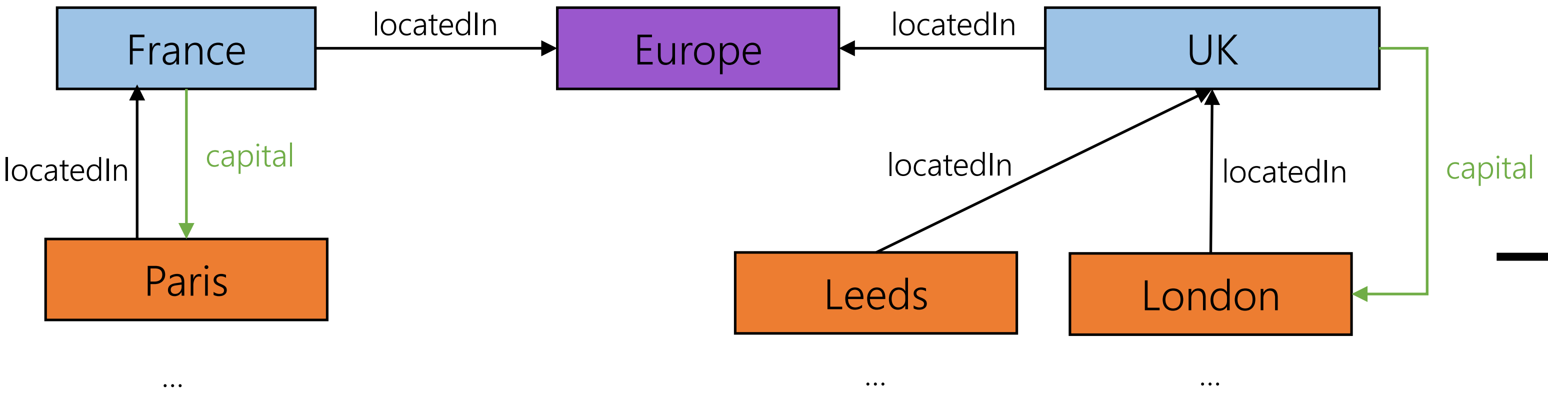


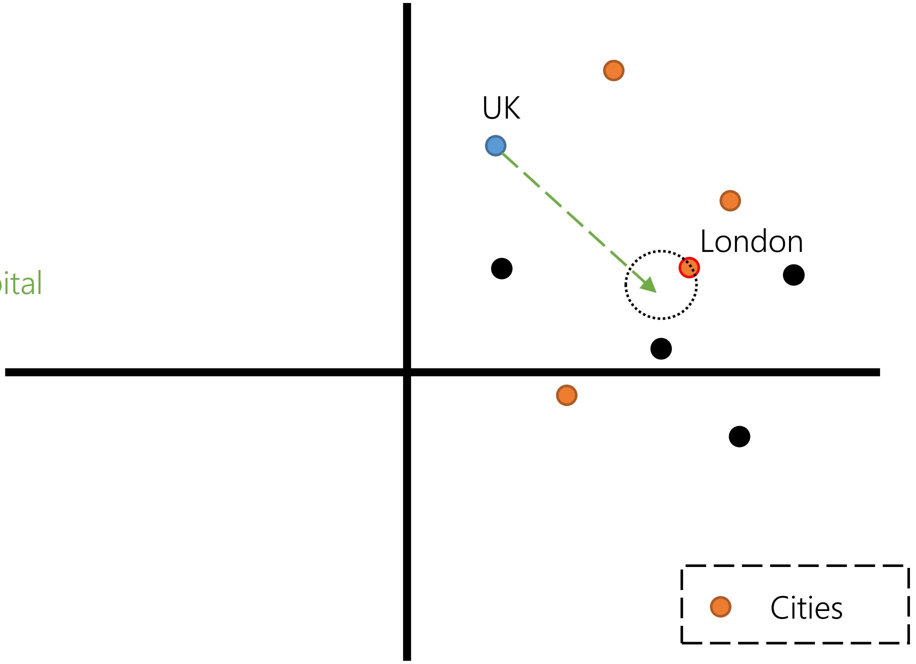


KGE Models: Applications

Generating KGE allows to use well-know **machine learning** models over KG (e.g., classification, clustering,…). However, we can also use them directly:

Knowledge Graph Completion

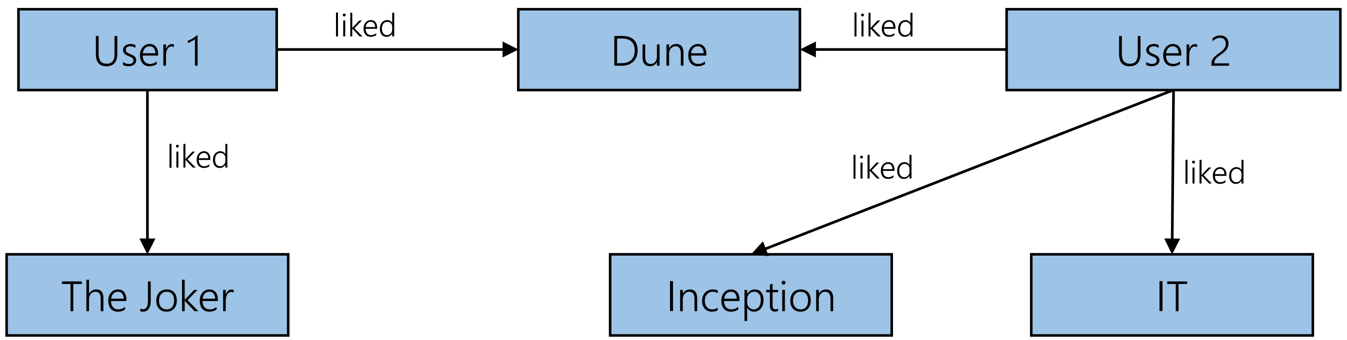


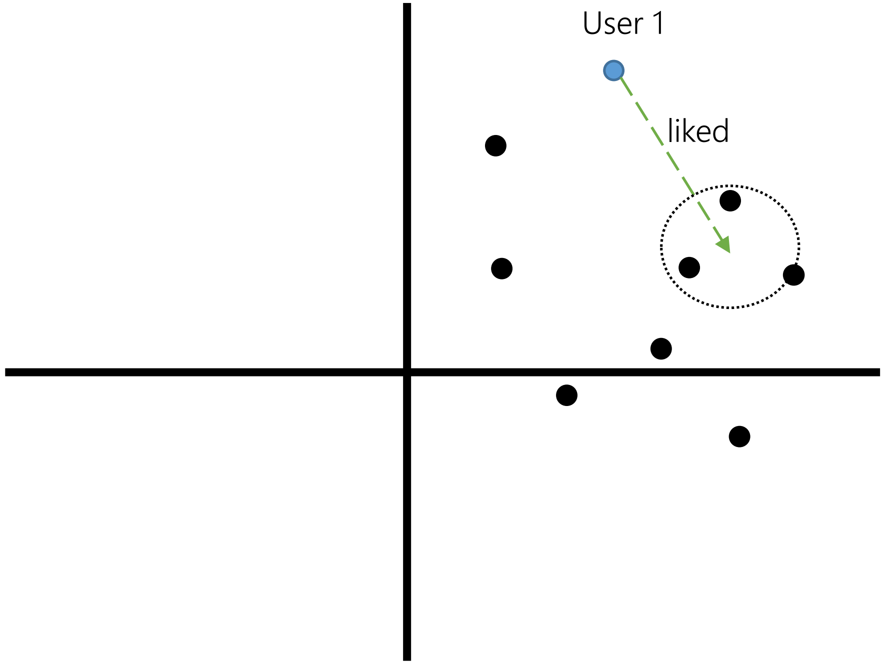


KGE Models: Applications

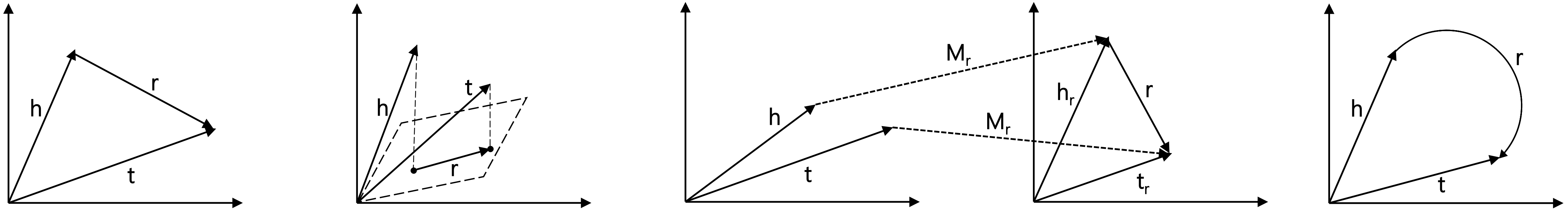
Generating KGE allows to use well-know **machine learning** models over KG (e.g., classification, clustering,…). However, we can also use them directly:

Recommender Systems





KGE Models: Problems



TransE TransH TransR RotatE

- There is no **distinction** between **ABOX** and **TBOX**.
- **Literals** are usually not considered.
- There is no general **best model**.

Discussion

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH