IT5429E-1-24 (24.1A01)(Fall 2024): Graph Analytics for Big Data

Week 5: Heterogeneous Graphs & Knowledge Graphs

Instructor: Thanh H. Nguyen

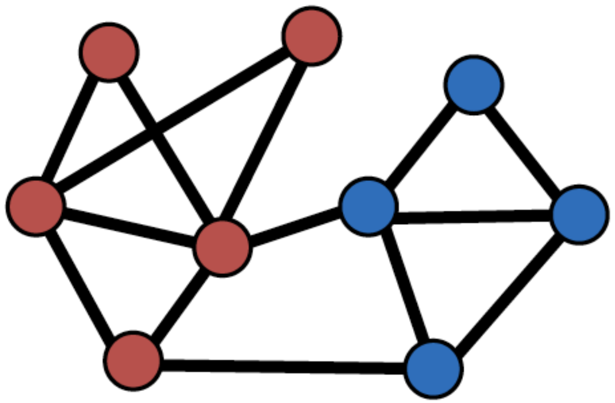
Many slides are adapted from https://web.stanford.edu/class/cs224w/

Heterogeneous Graphs

Heterogeneous Graphs

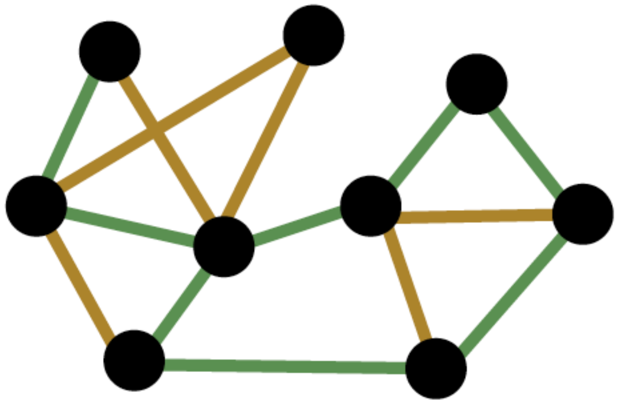
- So far we only handle graphs with one edge type
- How to handle graphs with multiple nodes or edge types (a.k.a heterogeneous graphs)?
- Goal: Learning with heterogeneous graphs
  - Relational GCNs
  - Design space for heterogeneous GNNs

Motivation



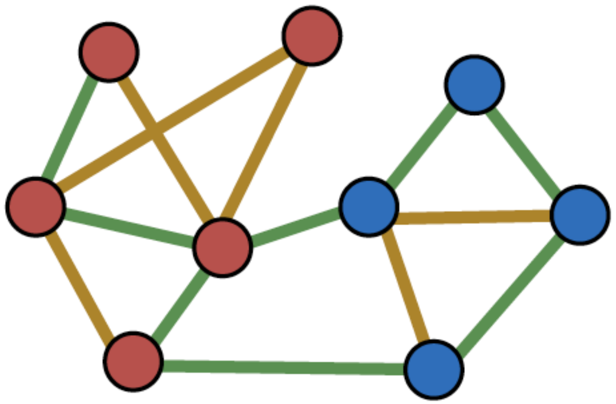
- Two types of nodes
  - Node type A: Paper nodes
  - Node type B: Author nodes

Motivation



- Two types of edges
  - Edge type A: Cite
  - Edge type B: Like

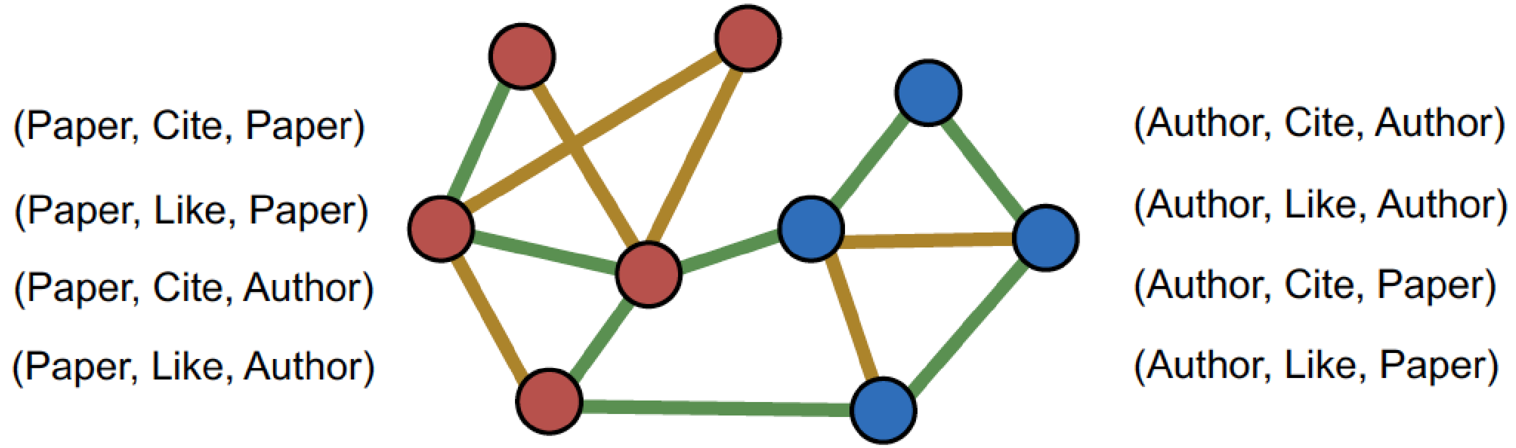
Motivation



- A graph could have multiple types of nodes and edges!
- 2 types of nodes + 2 types of edges.

Motivation

8 possible relation types

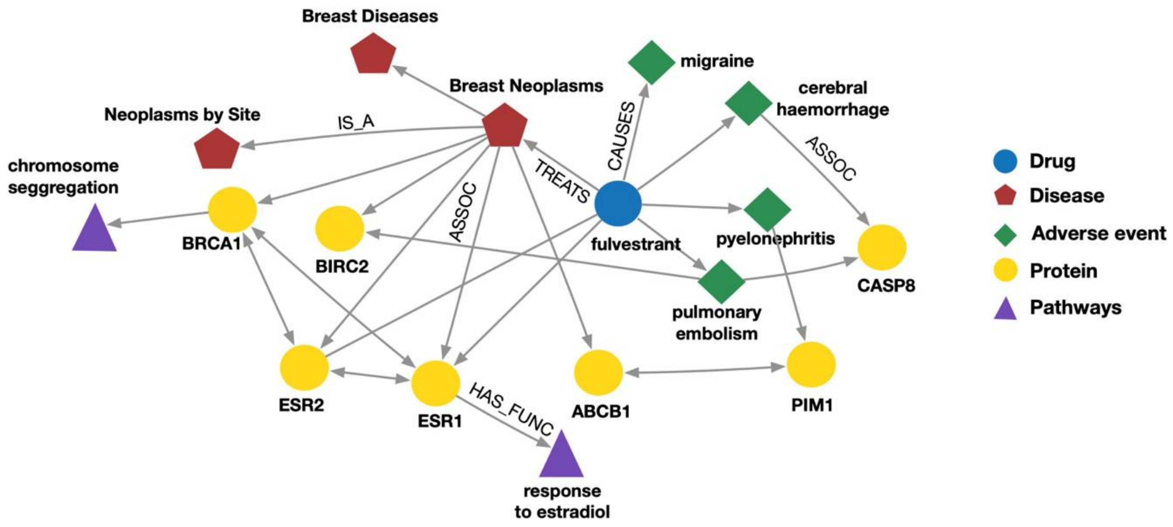


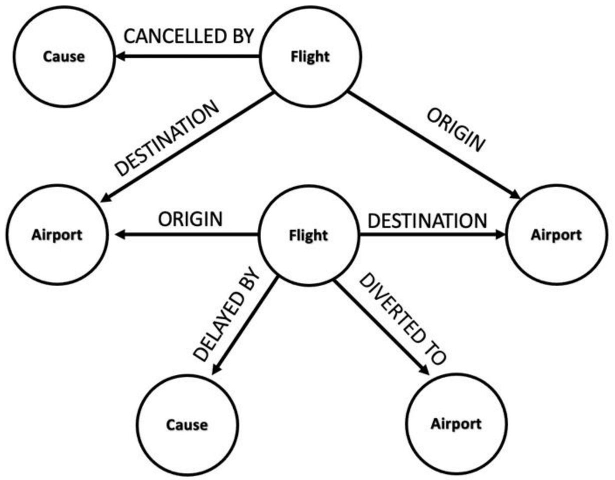
- Relation types: (node_start, edge, node_end)
  - We use relation type to describe an edge (as opposed to edge type)
  - Relation type better captures the interaction between nodes and edges

Heterogeneous Graphs

- A heterogeneous graph is defined as $G = (V, E, \tau, \phi)$
  - Nodes with node types $v \in V$
    - Node type for node v: $\tau(v)$
  - Edges with edge types $(u, v) \in E$
    - Edge type for edge (u, v): $\phi(u, v)$
    - Relation type for edge (u, v) is a tuple: $r(u, v) = \big(\tau(u), \phi(u, v), \tau(v)\big)$
- There are other definitions for heterogeneous graphs as well – describe graphs with node & edge types

Many Graphs are Heterogeneous Graphs





Biomedical Knowledge Graphs

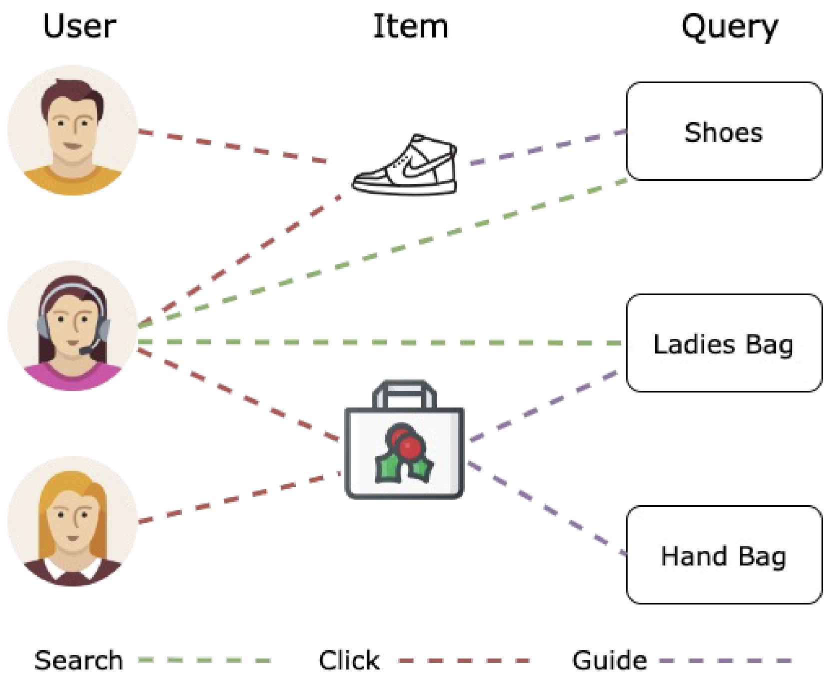
- Example node: Migraine
- Example relation: (fulvestrant, Treats, Breast Neoplasms)
- Example node type: Protein
- Example edge type: Causes

Event Graphs

- Example node: SFO
- Example relation: (UA689, Origin, LAX)
- Example node type: Flight
- Example edge type: Destination

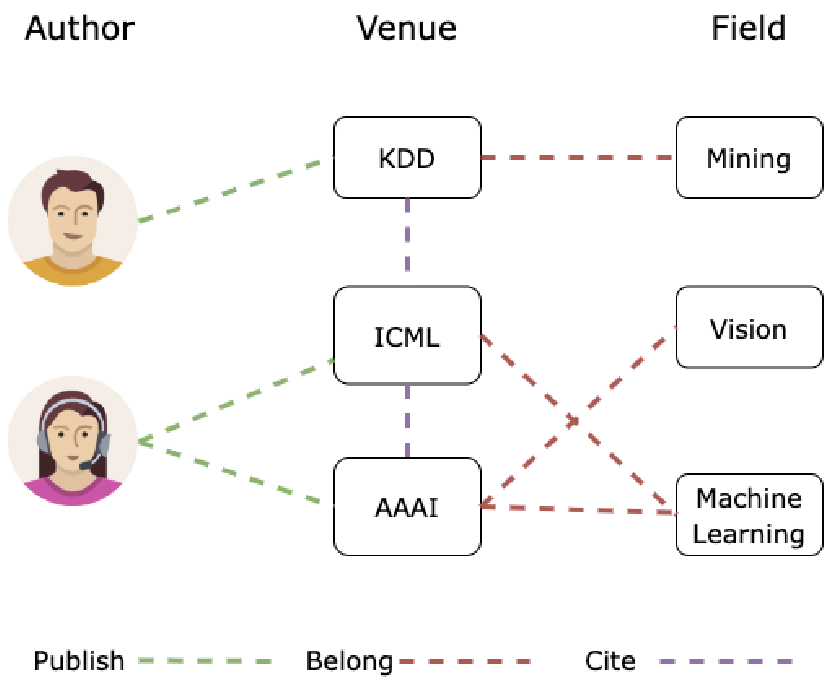
Many Graphs are Heterogeneous Graphs

- Example: E-Commerce Graph
  - Node types: User, Item, Query, Location, ...
  - Edge types: Purchase, Visit, Guide, Search, ...
  - Different node types' feature spaces can be different!



Many Graphs are Heterogeneous Graphs

- Example: Academic Graph
  - Node types: Author, Paper, Venue, Field, …
  - Edge types: Publish, Cite, …
  - Benchmark dataset: Microsoft Academic Graph



Discussion: Type or Feature

- Observation: We can also treat types of nodes and edges as features
  - Example: Add a one-hot indicator for nodes and edges
    - Append feature [1, 0] to each "author node"; Append feature [0, 1] to each "paper node"
    - Similarly, we can assign edge features to edges with different types
  - Then, a heterogeneous graph reduces to a standard graph
- When do we need a heterogeneous graph?

Discussion: Type or Feature

- When do we need a heterogeneous graph?
- Case 1: Different node/edge types have different shapes of features
  - An "author node" has 4-dim feature, a "paper node" has 5-dim feature
- Case 2: We know different relation types represent different types of interactions
  - (English, translate, French) and (English, translate, Chinese) require different models

Discussion: Heterogeneous

- Ultimately, heterogeneous graph is a more expressive graph representation
  - Captures different types of interactions between entities

- But it also comes with costs
  - More expensive (computation, storage)
  - More complex implementation

- There are many ways to convert a heterogeneous graph to a standard graph (that is, a homogeneous graph)

Relational GCN (RGCN)

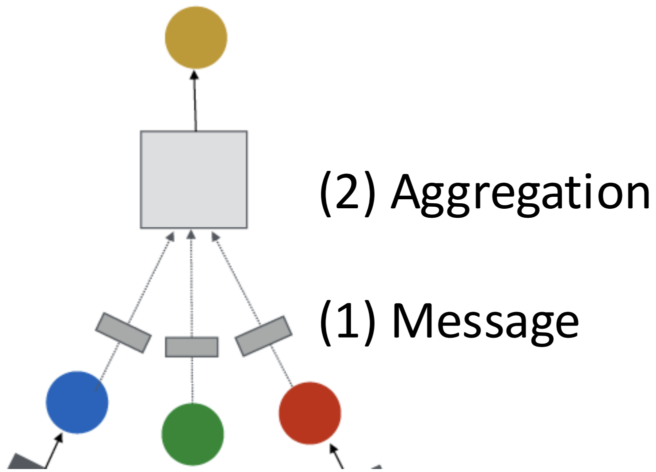
Recap: Classical GNN Layers: GCN

- (1) Graph Convolutional Networks (GCN)

$$\boxed{h_v^{(l)} = \sigma\left(W^{(l)} \sum_{u \in N(v)} \frac{h_u^{(l-1)}}{|N(v)|} \right)}$$

- How to write this as Message + Aggregation?

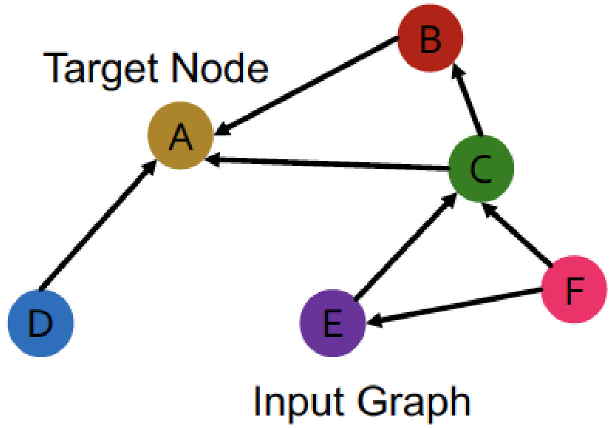
$$h_v^{(l)} = \sigma\left(\underbrace{\sum_{u \in N(v)}}_{\text{(2) Aggregation}} \overbrace{W^{(l)} \frac{h_u^{(l-1)}}{|N(v)|}}^{\text{(1) Message}} \right)$$



T. Kipf, M. Welling. [Semi-Supervised Classification with Graph Convolutional Networks](), ICLR 2017

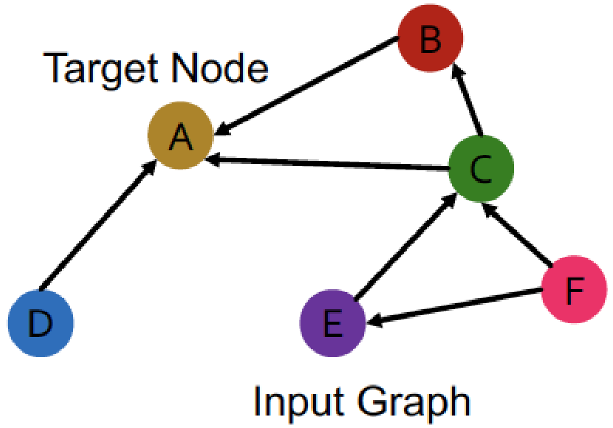
Relational GCN

- We will extend GCN to handle heterogeneous graphs with multiple edge/relation types
- We start with a directed graph with one relation
  - How do we run GCN and update the representation of the target node A on this graph?

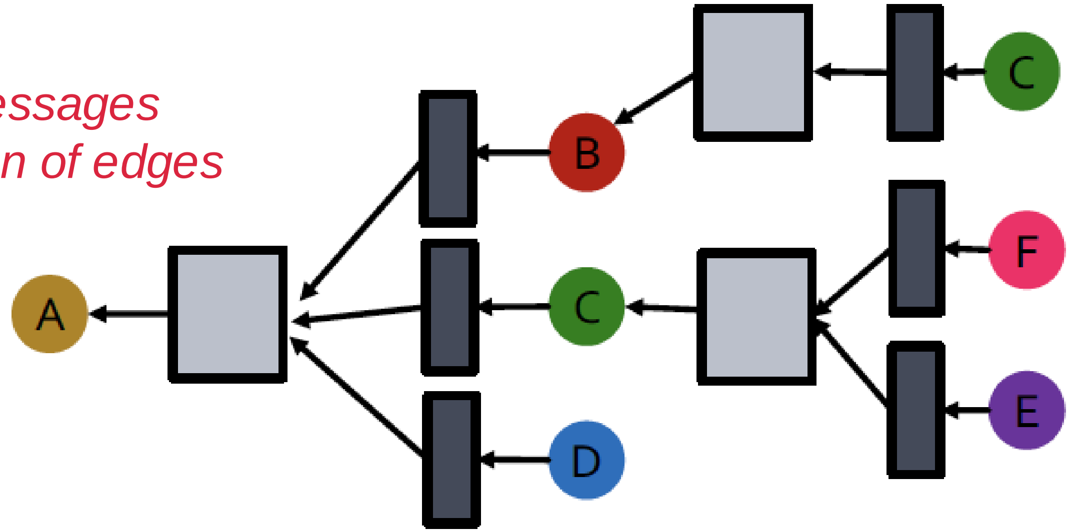


Relational GCN

- We will extend GCN to handle heterogeneous graphs with multiple edge/relation types
- We start with a directed graph with one relation
  - How do we run GCN and update the representation of the target node A on this graph?

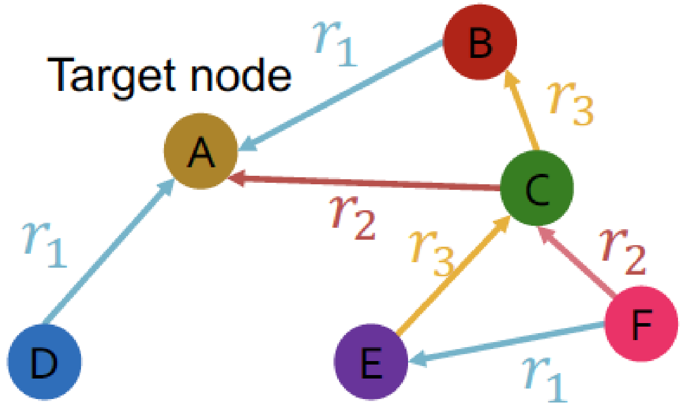


Only pass messages along direction of edges



Relational GCN (1)

- What if the graph has multiple relation types?



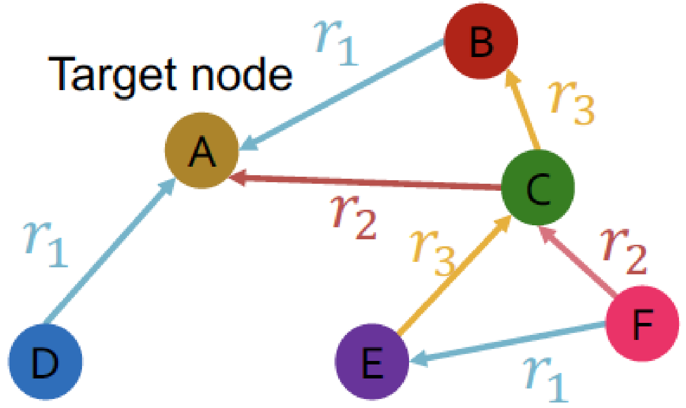
Input graph

Relational GCN (2)

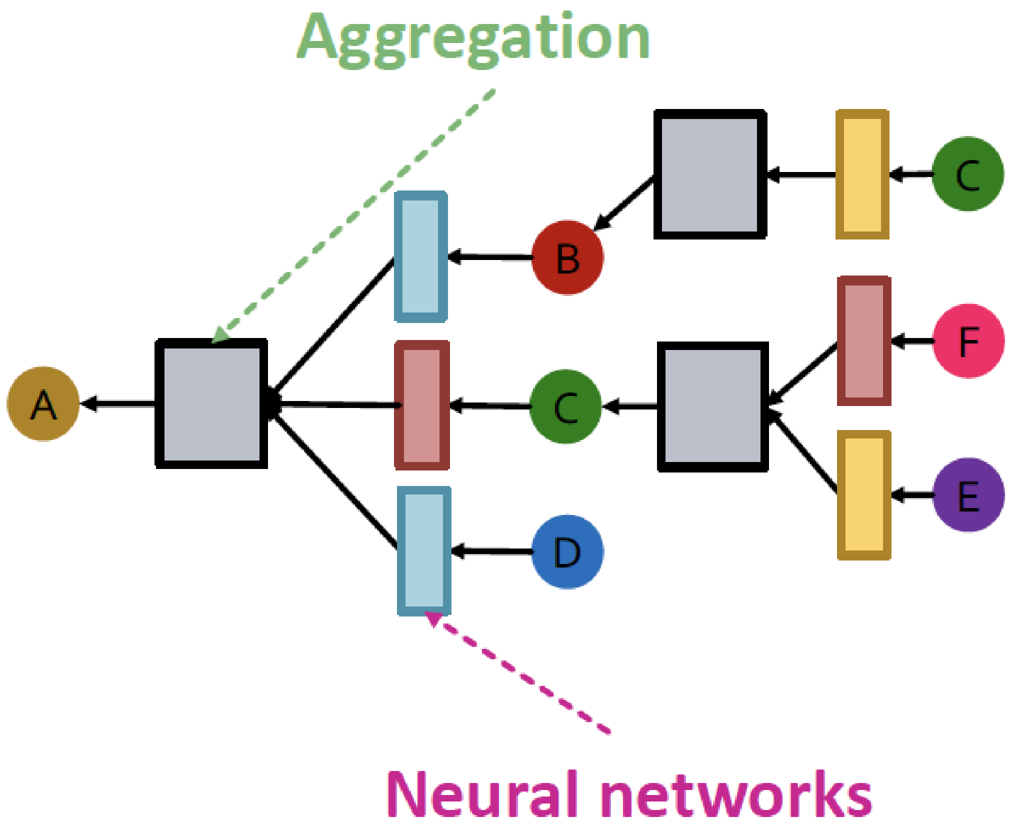
- What if the graph has multiple relation types?
- Use different neural network weights for different relation types.

Relational GCN (3)

- What if the graph has multiple relation types?
- Use different neural network weights for different relation types.

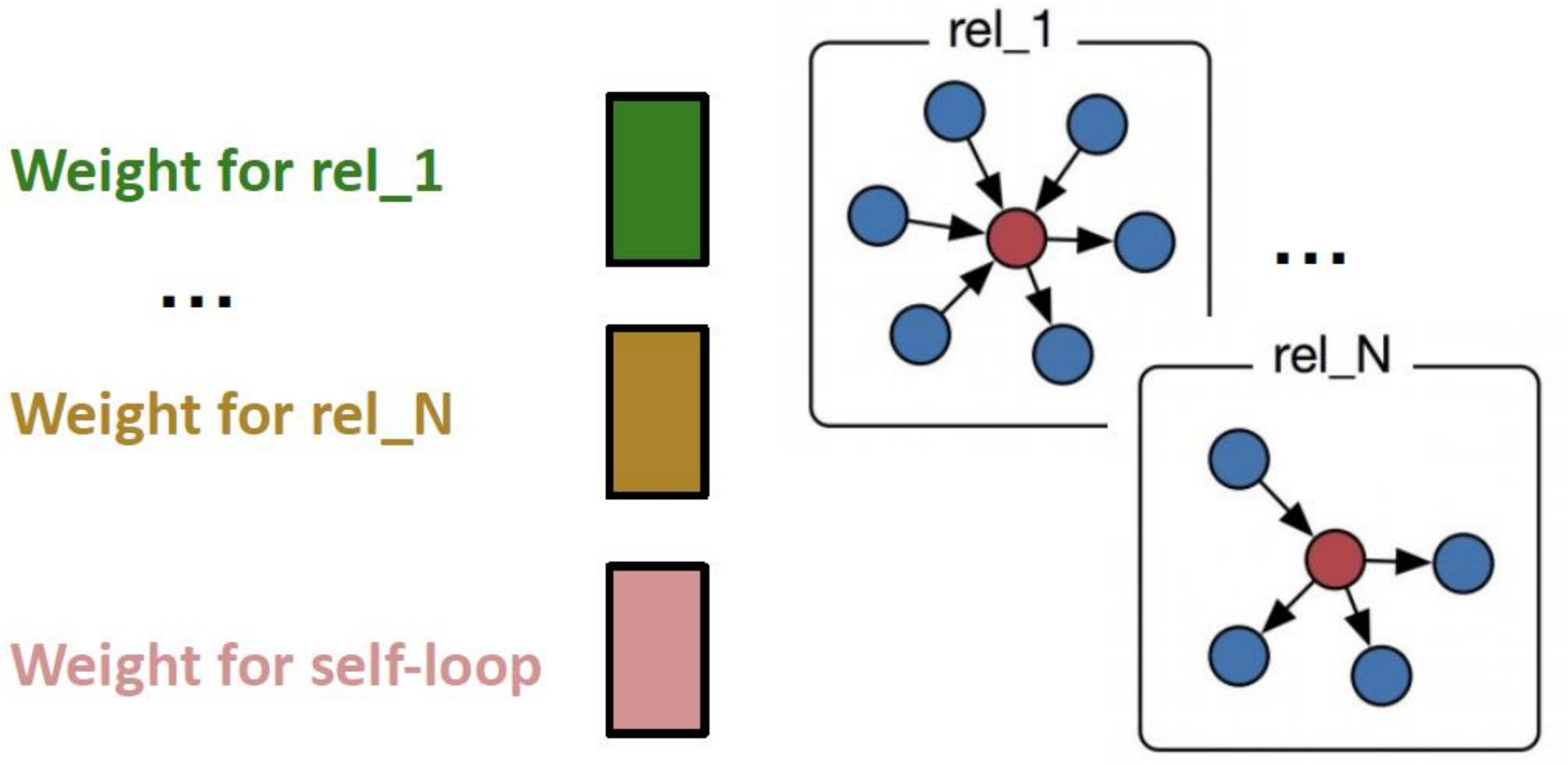


Input graph



Relational GCN (4)

- Introduce a set of neural networks for each relation type!



Relational GCN: Definition

- Relational GCN (RGCN):

$$h_v^{(l+1)} = \sigma\left(\sum_{r \in R} \sum_{u \in N_v^r} \frac{1}{c_{v,r}} W_r^{(l)} h_u^{(l)} + W_0^{(l)} h_v^{(l)}\right)$$

Normalized by node degree of the relation $c_{v,r} = |N_v^r|$

- How to write this as Message + Aggregation?
- Message:
  - Each neighbor of a given relation: $m_{u,r}^{(l)} = \frac{1}{c_{v,r}} W_r^{(l)} h_u^{(l)}$
  - Self-loop: $m_v^{(l)} = W_0^{(l)} h_v^{(l)}$
- Aggregation:
  - Sum over messages from neighbors and self-loop, then apply activation
  - $h_v^{(l+1)} = \sigma\left(Sum\left(\left\{m_{u,r}^{(l)}, u \in N(v)\right\} \cup \left\{m_v^{(l)}\right\}\right)\right)$

RGCN: Scalablity

- Each relation has L matrices: $W_r^{(1)}, W_r^{(2)}, \cdots, W_r^{(L)}$
- The size of each $W_r^{(l)}$ is $d^{l+1} \times d^{l}$

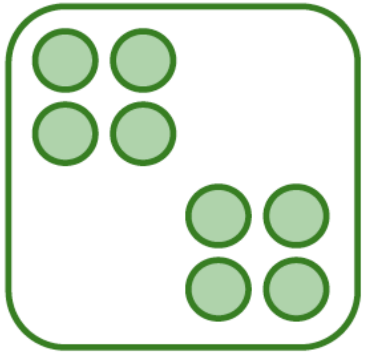
d^l is the hidden dimension in layer l

- Rapid growth of the number of parameters w.r.t number of relations!
  - Overfitting becomes an issue

- Two methods to regularize the weights $W_r^{(l)}$
  - (1) Use block diagonal matrices
  - (2) Basis/Dictionary learning

(1) Block Diagonal Matrices

- Key insight: make the weights sparse!
- Use block diagonal matrices for W_r

$\mathbf{W}_r =$ 

Limitation: only nearby neurons/dimensions can interact through W

- If use B low-dimensional matrices, then # param reduces from $d^{l+1} \times d^{l}$ to $\mathrm{B} \times \frac{d^{l+1}}{B} \times \frac{d^{l}}{B}$

(2) Basis Learning

- Key insight: Share weights across different relations!
- Represent the matrix of each relation as a linear combination of basis transformations

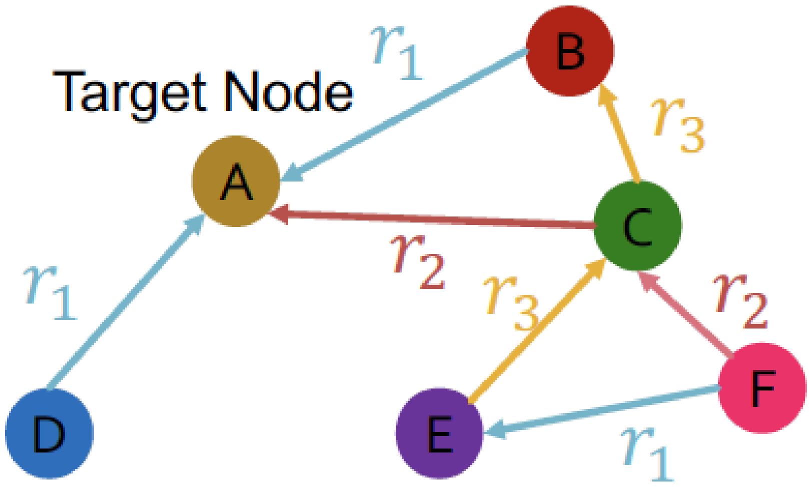
$$\boxed{W_r = \sum_{b=1}^{B} a_{rb} \times V_b}$$

  - V_b are the basis matrices, shared across all relations
  - a_{rb}: the importance weight of matrix V_b

- Now each relation only needs to learn $\{a_{rb}\}_{b=1}^{B}$ which is B scalars

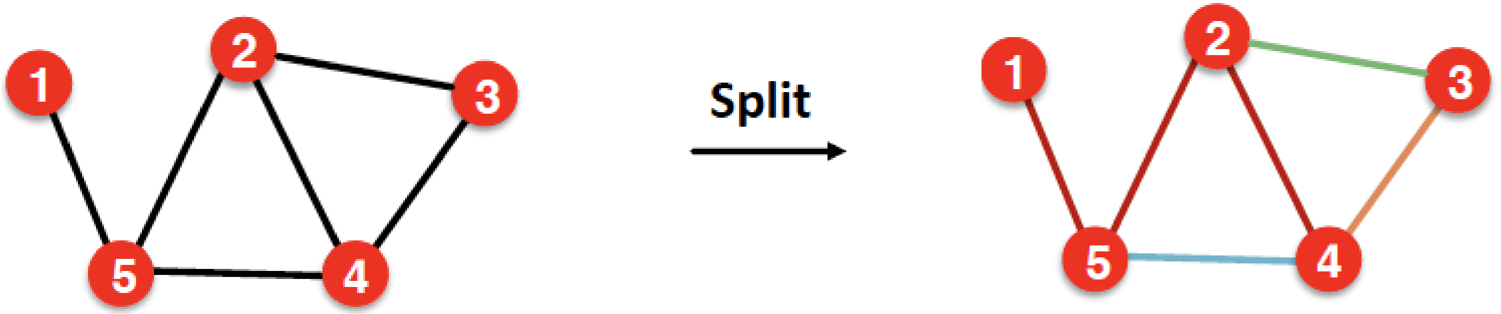
Example: Entity/Node Classification

- Goal: Predict the label of a given node
- RGCN uses the representation of the final layer:
  - If we predict the class of node ***A*** from ***k*** classes
  - Take the final layer (prediction head): $h_A^{(L)} \in \mathbb{R}^k$, each item in $h_A^{(L)}$ represents the probability of that class.



Example: Link Prediction

- Link prediction split



The original graph

Split Graph with 4 categories of edges

Training message edges for r_1
Training supervision edges for r_1
Validation edges for r_1
Test edges for r_1

…

Training message edges for r_n
Training supervision edges for r_n
Validation edges for r_n
Test edges for r_n

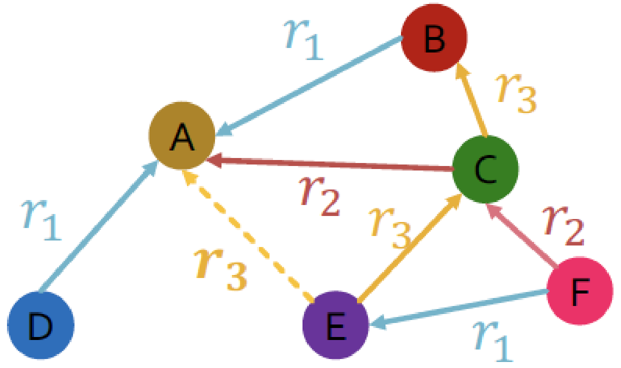
Training message edges
Training supervision edges
Validation edges
Test edges

Every edge also has a relation type, this is independent of the 4 categories.

In a heterogeneous graph, the homogeneous graphs formed by every single relation also have the 4 splits.

Example: Link Prediction (1)

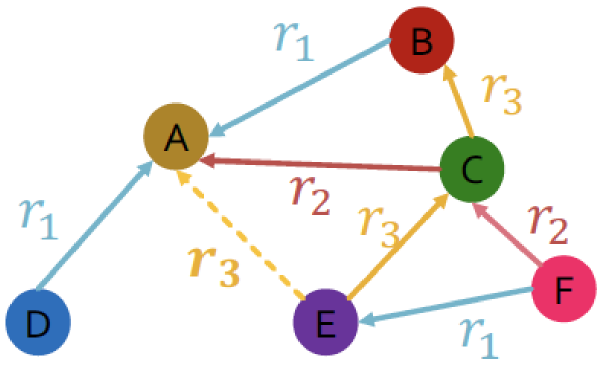
- Assume (E, r_3, A) is training supervision edge, all other edges are training message edges
- Use RGCN to score (E, r_3, A)!
  - Take the final layer of E and A: $h_E^{(L)}$ and $h_A^{(L)} \in \mathbb{R}^d$
  - Relation-specific score function: $f_r: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
    - One example: $f_{r_3}(h_E, h_A) = h_E^T W_{r_3} h_A,\ W_{r_3} \in \mathbb{R}^{d \times d}$



Input Graph

Example: Link Prediction (2)

- Training



Input Graph

1. Use RGCN to score the training supervision edge (E, r_3, A)
2. Create a negative edge by perturbing the supervision edge (E, r_3, B)
   - Corrupt the tail of (E, r_3, A)
     - E.g., $(E, r_3, B), (E, r_3, D)$

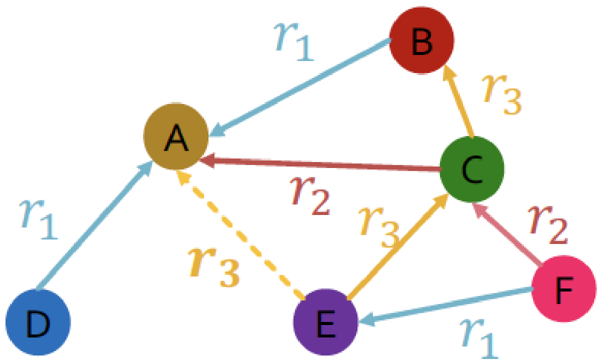
training supervision edges: $(\mathrm{E}, \mathrm{r}_3, \mathrm{A})$
training message edges: all the rest existing edges (solid lines)

(1) Use training message edges to predict training supervision edges

Note the negative edges should NOT belong to training message edges or training supervision edges!
e.g., (E, r_3, C) is NOT a negative edge

Example: Link Prediction (3)

- Training



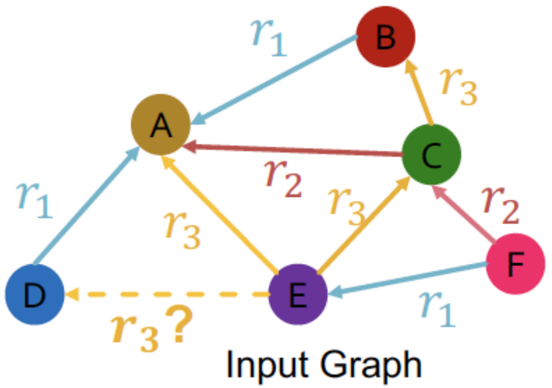
Input Graph

1. Use RGCN to score the training supervision edge (E, r_3, A)
   Create a negative edge by perturbing the supervision edge (E, r_3, B)
2. Use GNN model to score negative edge
3. Optimize a standard cross entropy loss
   1. Maximize the score of training supervision edge
   2. Minimize the score of negative edge

$$l = -\log \sigma\left(f_{r_3}(h_E, h_A)\right) - \log\left(1 - \sigma\left(f_{r_3}(h_E, h_B)\right)\right)$$

Example: Link Prediction (4)

- Evaluation:
  - Validation time as an example, same at the test time



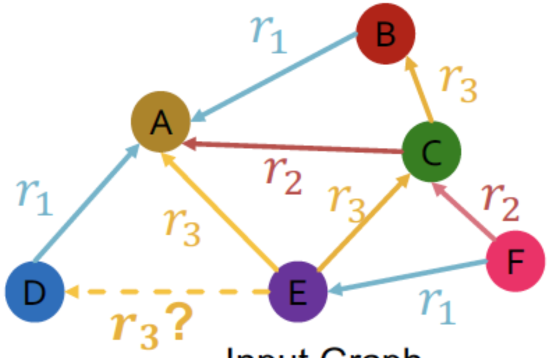
Evaluate how the model can predict the validation edges with the relation types.
Let's predict validation edge (E, r_3, D)
Intuition: the score of $\boldsymbol{E}$, (E, r_3, D) should be higher than all (E, r_3, v) where (E, r_3, v) is NOT in the training message edges and training supervision edges, e.g., (E, r_3, B)

validation edges: (E, r_3, D)
training message edges & training supervision edges: all existing edges (solid lines)

(2) At validation time:
Use training message edges & training supervision edges to predict validation edges

Example: Link Prediction (5)

- Evaluation:
  - Validation time as an example, same at the test time



Input Graph

Evaluate how the model can predict the validation edges with the relation types.

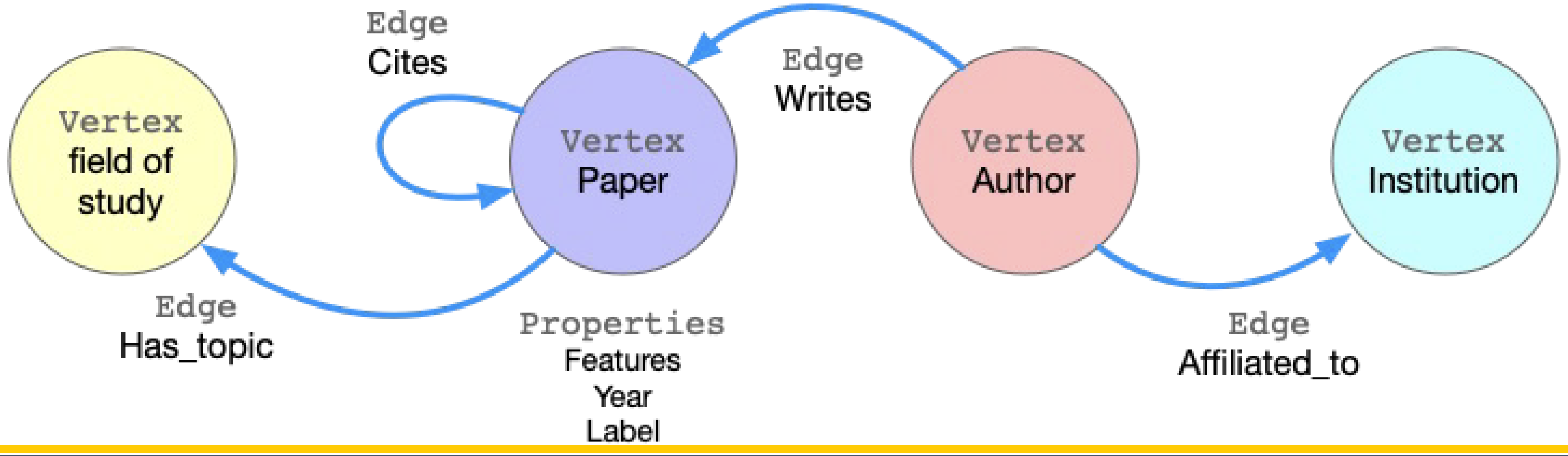
Let's predict validation edge (E, r_3, D)

Intuition: the score of $\boldsymbol{E}$, (E, r_3, D) should be higher than all (E, r_3, v) where (E, r_3, v) is NOT in the training message edges and training supervision edges, e.g., (E, r_3, B)

1. *Calculate the score of* (E, r_3, D)
2. *Calculate the score of all the negative edges:* $\{(E, r_3, v) \mid v \in \{B, F\}\}$, *since* (E, r_3, A), (E, r_3, C) *belong to training message edges & training supervision edges*
3. *Obtain the ranking* $\boldsymbol{RK}$ *of* (E, r_3, D).
4. *Calculate metrics*
   1. *Hits@k:* $1[RK \leq k]$. *Higher is better*
   2. *Reciprocal Rank:* $\frac{1}{RK}$. *Higher is better*

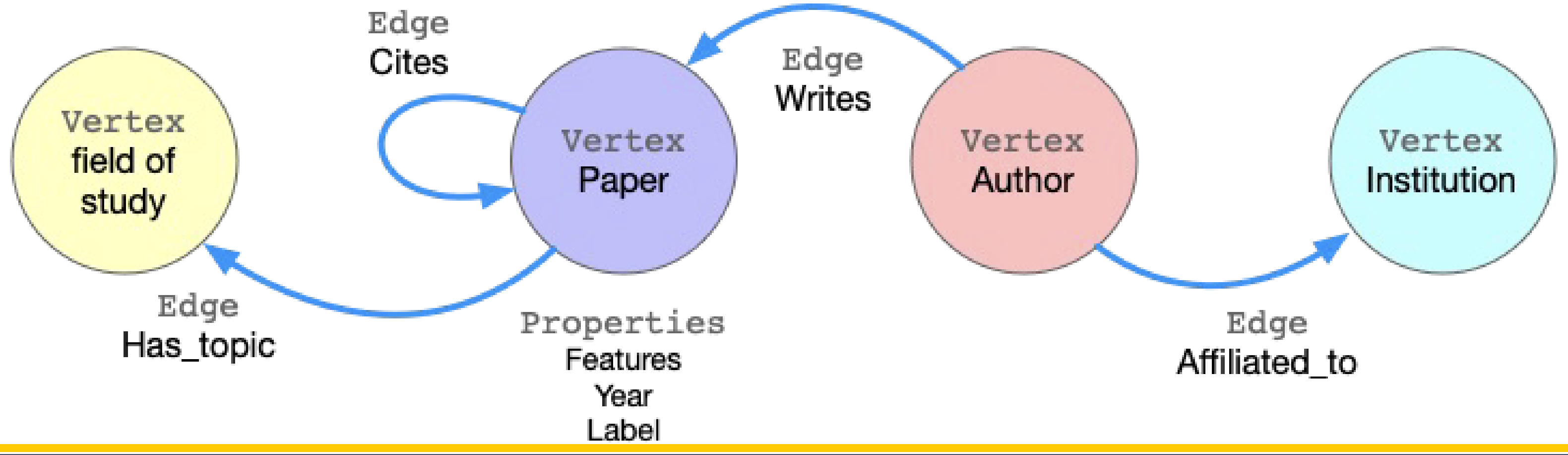
Benchmark for Heterogeneous Graphs (1)

- Benchmark dataset
  - ogbn-mag from Microsoft Academic Graph (MAG)
- Four (4) types of entities
  - Papers: 736k nodes
  - Authors: 1.1m nodes
  - Institutions: 9k nodes
  - Fields of study: 60k nodes



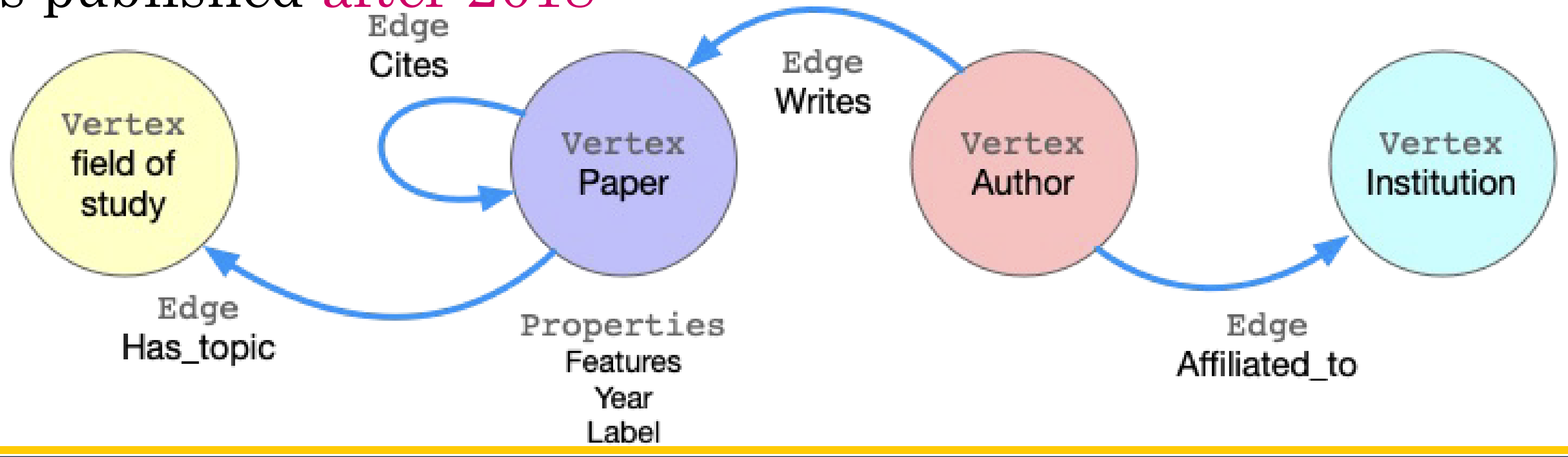
Benchmark for Heterogeneous Graphs (2)

- Benchmark dataset
  - ogbn-mag from Microsoft Academic Graph (MAG)
- Four (4) directed relations
  - An author is "affiliated with" an institution
  - An author "writes" a paper
  - A paper "cites" a paper
  - A paper "has a topic of" a field of study



Benchmark for Heterogeneous Graphs (3)

- Prediction task
  - Each paper has a 128-dimensional word2vec feature vector
  - Given the content, references, authors, and author affiliations from ogbn-mag, predict the venue of each paper
  - 349-class classification problem due to 349 venues considered
- Time-based dataset splitting
  - Training set: papers published before 2018
  - Test set: papers published after 2018

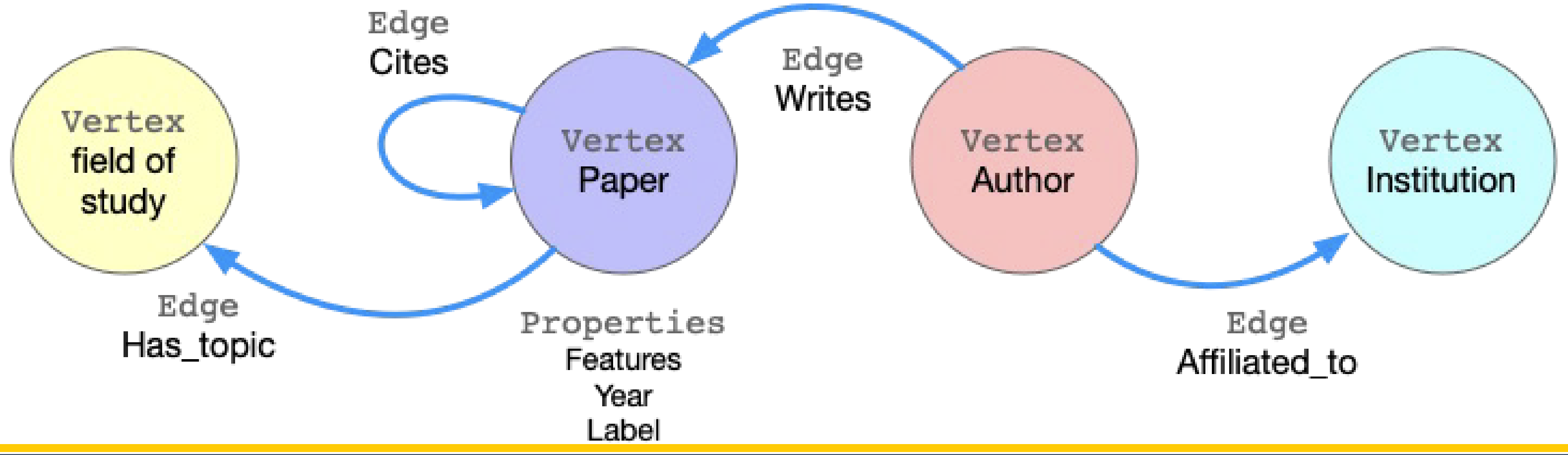


Benchmark for Heterogeneous Graphs (4)

- Benchmark results

Rank	Method	Ext. data	Test Accuracy	Validation Accuracy	Contact	References	#Params	Hardware	Date
1	**SeHGNN (ComplEx embs)**	No	0.5719 ± 0.0012	0.5917 ± 0.0009	Xiaocheng Yang (ICT-GIMLab)	Paper, Code	8,371,231	NVIDIA Tesla T4 (15 GB)	Jul 7, 2022
21	NeighborSampling (R-GCN aggr)	No	0.4678 ± 0.0067	0.4761 ± 0.0068	Matthias Fey – OGB team	Paper, Code	154,366,772	GeForce RTX 2080 (11GB GPU)	Jun 26, 2020

- SOTA method: SeHGNN
  - ComplEx (study later) + Simplified GCN

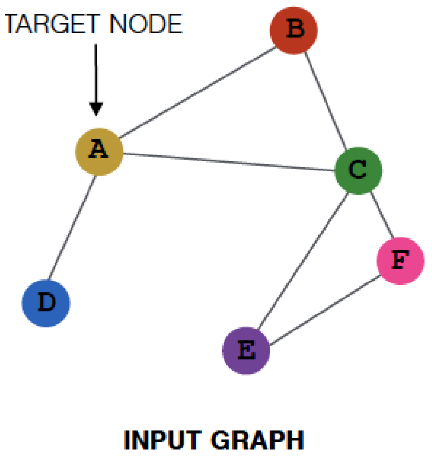


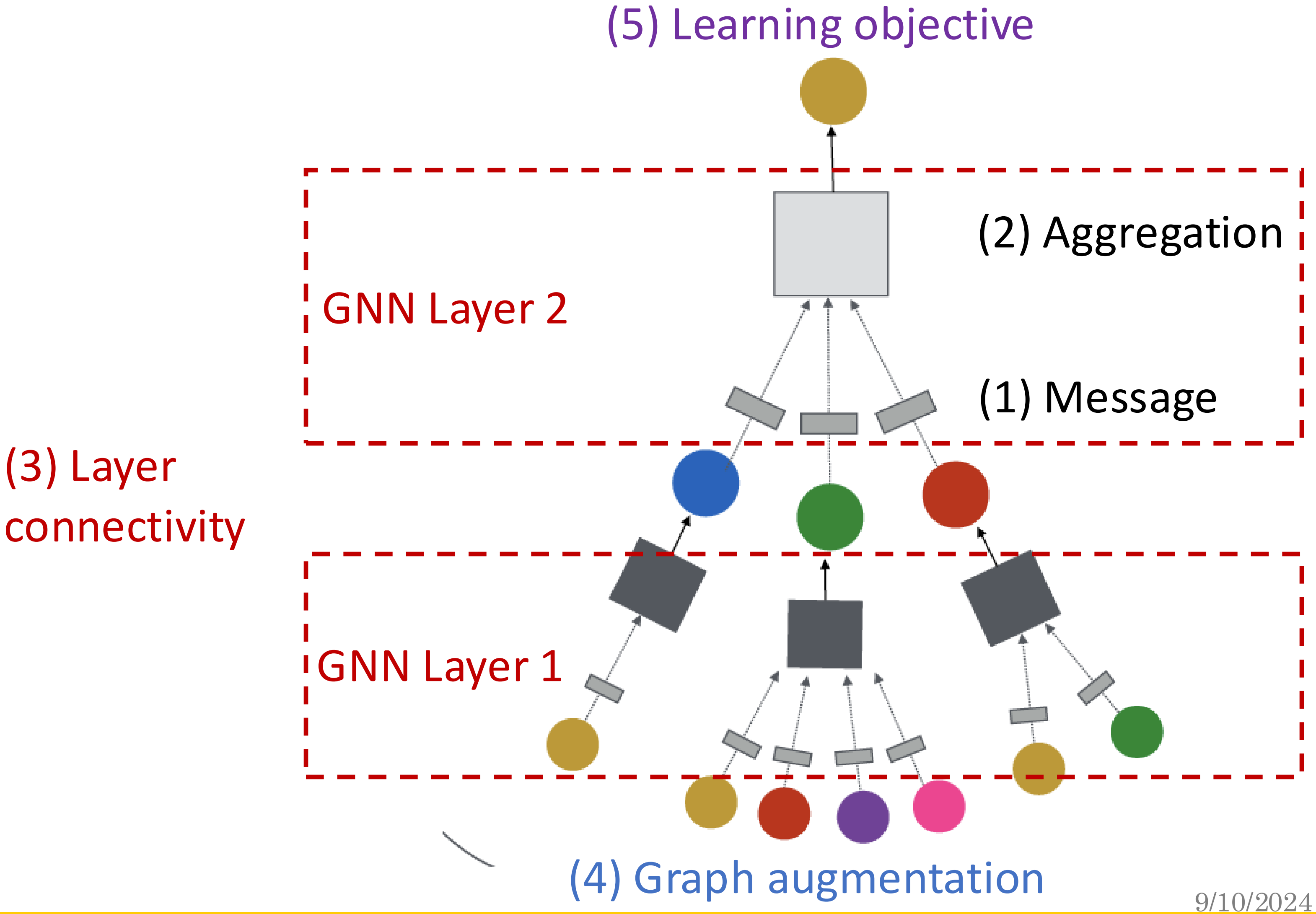
Summary of RGCN

- Relational GCN, a graph neural network for heterogeneous graphs
- Can perform entity classification as well as link prediction tasks.
- Ideas can easily be extended into RGNN (RGraphSAGE, RGAT, etc.)
- Benchmark: ogbn-mag from Microsoft Academic Graph, to predict paper venues

Design Space of Heterogeneous GNNs

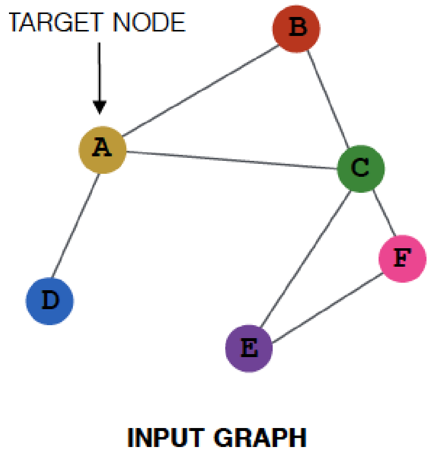
Recap: A General GNN Framework

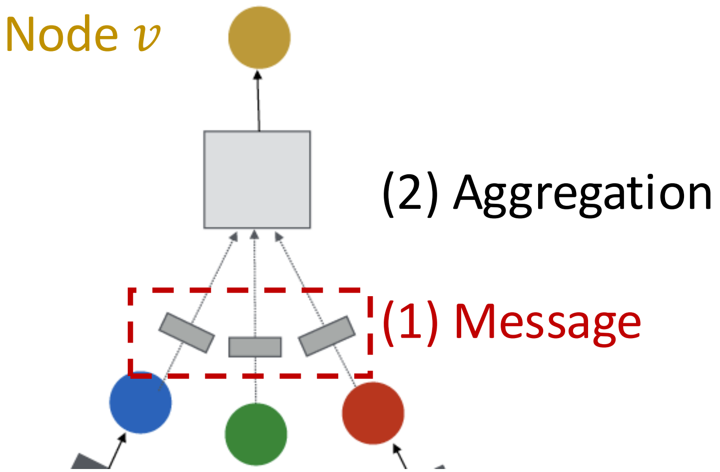




Recap: Message Computation

- (1) Message computation
  - **Message function**: $m_u^{(l)} = MSG^{(l)}\left(h_u^{(l-1)}\right)$
    - **Intuition**: Each node will create a message, which will be sent to other nodes later
    - **Example**: A linear layer $m_u^{(l)} = W^{(l)} h_u^{(l-1)}$
      - Multiply node features with weight matrix $m_u^{(l)}$





Heterogeneous Message

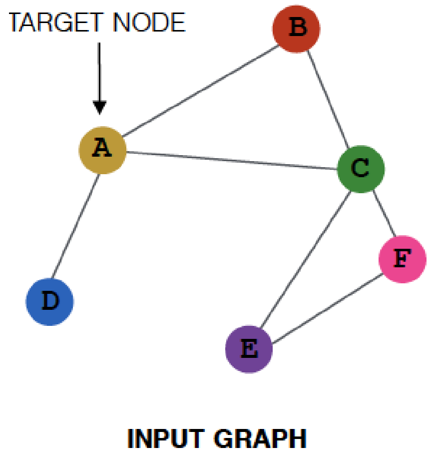
- (1) Heterogeneous message computation
  - **Message function**: $m_u^{(l)} = MSG_r^{(l)}\left(h_u^{(l-1)}\right)$
  - **Observation**: A node could receive multiple types of messages. Num of message type = Num of relation type
  - **Idea**: Create a different message function for each relation type
    - $m_u^{(l)} = MSG_r^{(l)}\left(h_u^{(l-1)}\right)$, $r = (u, e, v)$ is the relation type between node u that sends the message, edge type e, and node v that receive the message
  - **Example**: A Linear layer $m_u^{(l)} = W_r^{(l)} h_u^{(l-1)}$

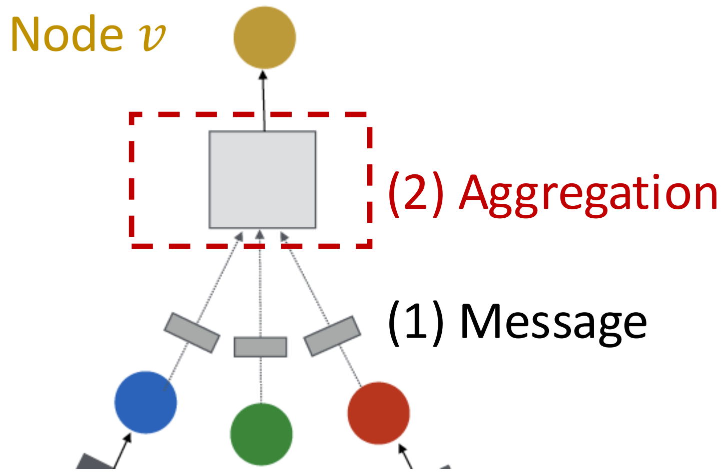
Recap: Message Aggregation

- (2) Aggregation
  - **Intuition**: Node v will aggregate messages from its neighbors u

$$h_v^{(l)} = AGG^{(l)}\left(\left\{m_u^{(l)}, u \in N(v)\right\}\right)$$

  - **Example:** AGG() can be Sum(), Mean(), or Max() aggregator



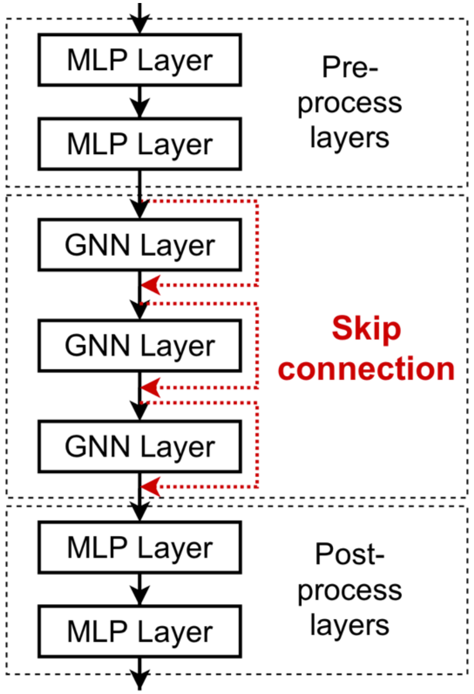


Heterogeneous Aggregation

- (2) Heterogeneous Aggregation
  - **Observation**: Each node could receive multiple types of messages from its neighbors, and multiple neighbors may belong to each message type.
  - **Idea**: We can define a 2-stage message passing
    - $h_v^{(l)} = AGG_{all}^{(l)}\left(AGG_r^{(l)}\left(\left\{m_u^{(l)}, u \in N_r(v)\right\}\right)\right)$
    - Given all the messages sent to a node
    - Within each message type, aggregate the messages that belongs to the edge type with $AGG_r^{(l)}$
    - Aggregate across the edge types with $AGG_{all}^{(l)}$
  - **Example**: $h_v^{(l)} = Concat\left(Sum\left(\left\{m_u^{(l)}, u \in N_r(v)\right\}\right)\right)$

Recap: Layer Connectivity

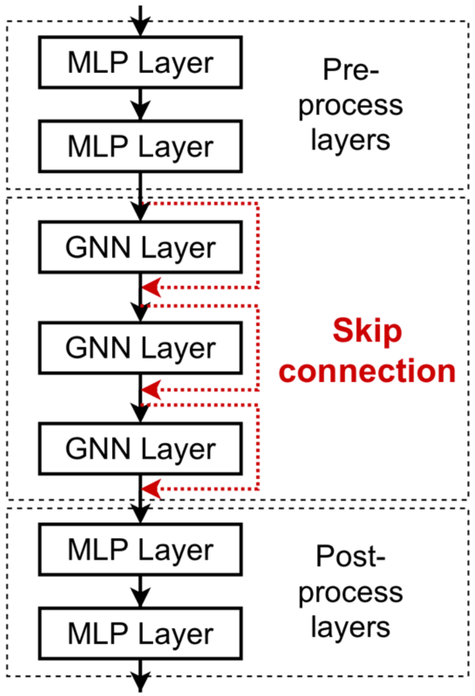
- (3) Layer connectivity
  - Add skip connections, pre/post-process layers



- Pre-processing layers: Important when encoding node features is necessary. E.g., when nodes represent images/text
- Post-processing layers: Important when reasoning / transformation over node embeddings are needed. E.g., graph classification, knowledge graphs
- In practice, adding these layers works great!

Heterogeneous GNN Layers

- Heterogeneous pre/post-process layers:
  - MLP layers with respect to each node type
  - Since the output of GNN are node embeddings
  - $h_v^{(l)} = MLP_{T(v)}\left(h^{(l)}(v)\right)$
- Other successful GNN designs are also encouraged for heterogeneous GNNs: skip connections, batch/layer normalization, …



Recap: Graph Manipulation

- Graph Feature manipulation
  - The input graph lacks features → feature augmentation

- Graph Structure manipulation
  - The graph is too sparse → Add virtual nodes / edges
  - The graph is too dense → Sample neighbors when doing message passing
  - The graph is too large → Sample subgraphs to compute embeddings
    - Will cover later in lecture: Scaling up GNNs

Heterogeneous Graph Manipulation

- Graph Feature manipulation
  - Common options: compute graph statistics (e.g., node degree) within each relation type, or across the full graph (ignoring the relation types)

- Graph Structure manipulation
  - Neighbor and subgraph sampling are also common for heterogeneous graphs.
  - 2 Common options: sampling within each relation type (ensure neighbors from each type are covered), or sample across the full graph

Recap: GNN Prediction Heads

- Node-level prediction
  - $\hat{y}_v = Head_{node}\left(h_v^{(L)}\right) = W^{(H)} h_v^{(L)}$
- Edge-level prediction
  - $\hat{y}_{uv} = Head_{edge}\left(h_u^{(L)}, h_v^{(L)}\right) = Linear\left(Concat\left(h_u^{(L)}, h_v^{(L)}\right)\right)$
- Graph-level prediction
  - $\hat{y}_G = Head_{graph}\left(\left\{h_v^{(L)} \in \mathbb{R}^d, \forall v \in G\right\}\right)$

Heterogeneous Prediction Heads

- Node-level prediction
  - $\hat{y}_v = Head_{node,T(v)}\left(h_v^{(L)}\right) = W_{T(v)}^{(H)} h_v^{(L)}$
- Edge-level prediction
  - $\hat{y}_{uv} = Head_{edge,r}\left(h_u^{(L)}, h_v^{(L)}\right) = Linear_r\left(Concat\left(h_u^{(L)}, h_v^{(L)}\right)\right)$
- Graph-level prediction
  - $\hat{y}_G = AGG\left(Head_{graph,i}\left(\left\{h_v^{(L)} \in \mathbb{R}^d, \forall T(v) = i\right\}\right)\right)$

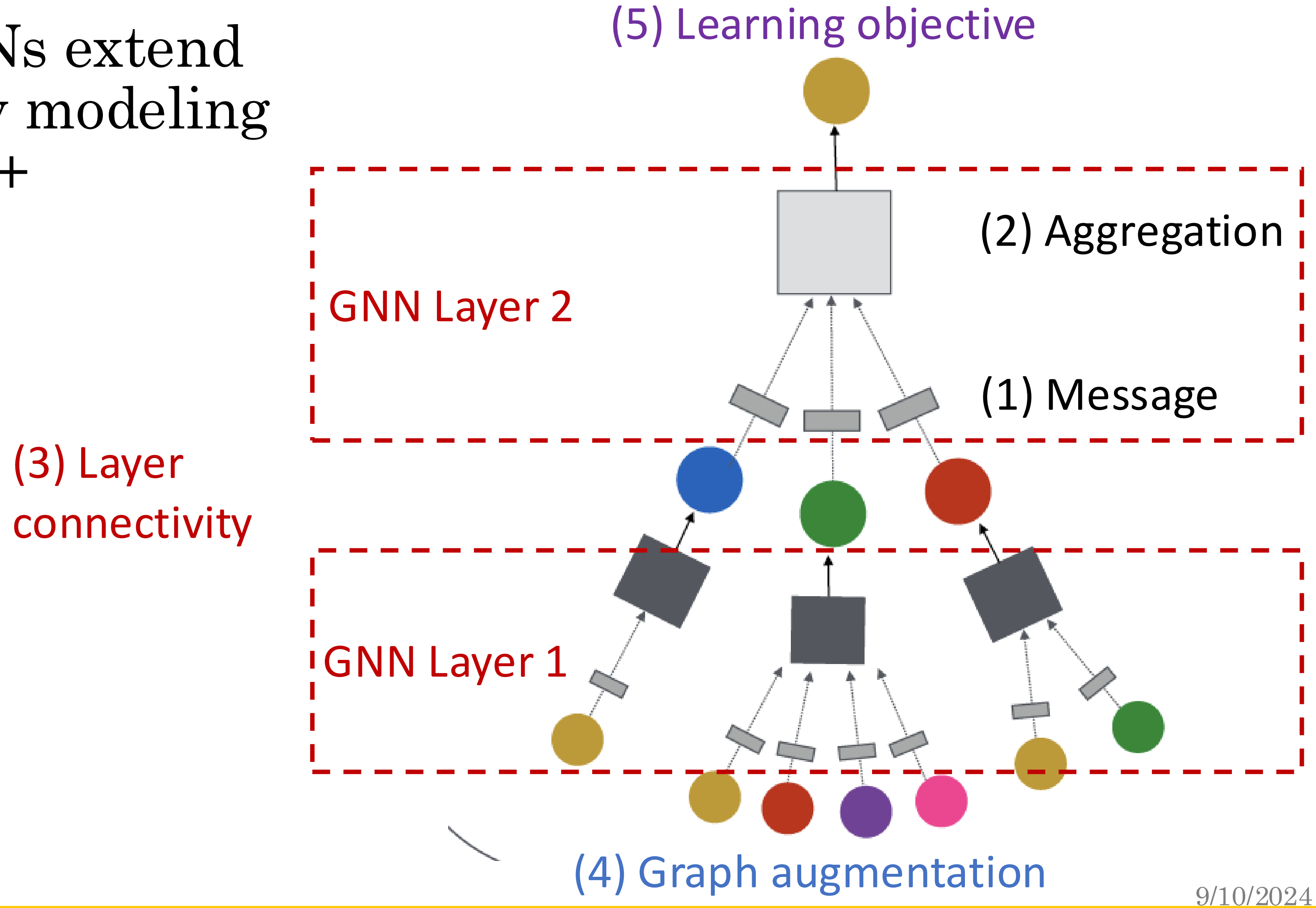
Summary

- Heterogeneous graphs: graphs with multiple nodes or edge types
  - Key concept: relation type (node_s, edge, node_e)
  - Be aware that we don't always need heterogeneous graphs

- Learning with heterogeneous graphs
  - Key idea: separately model each relation type
  - Relational GCNs
  - Design space for heterogeneous GNNs

Summary: Heterogeneous GNN

- Heterogeneous GNNs extend GNNs by separately modeling node/relation types + additional AGG



Break Time!

Knowledge Graph Embeddings

Knowledge Graph (KG)

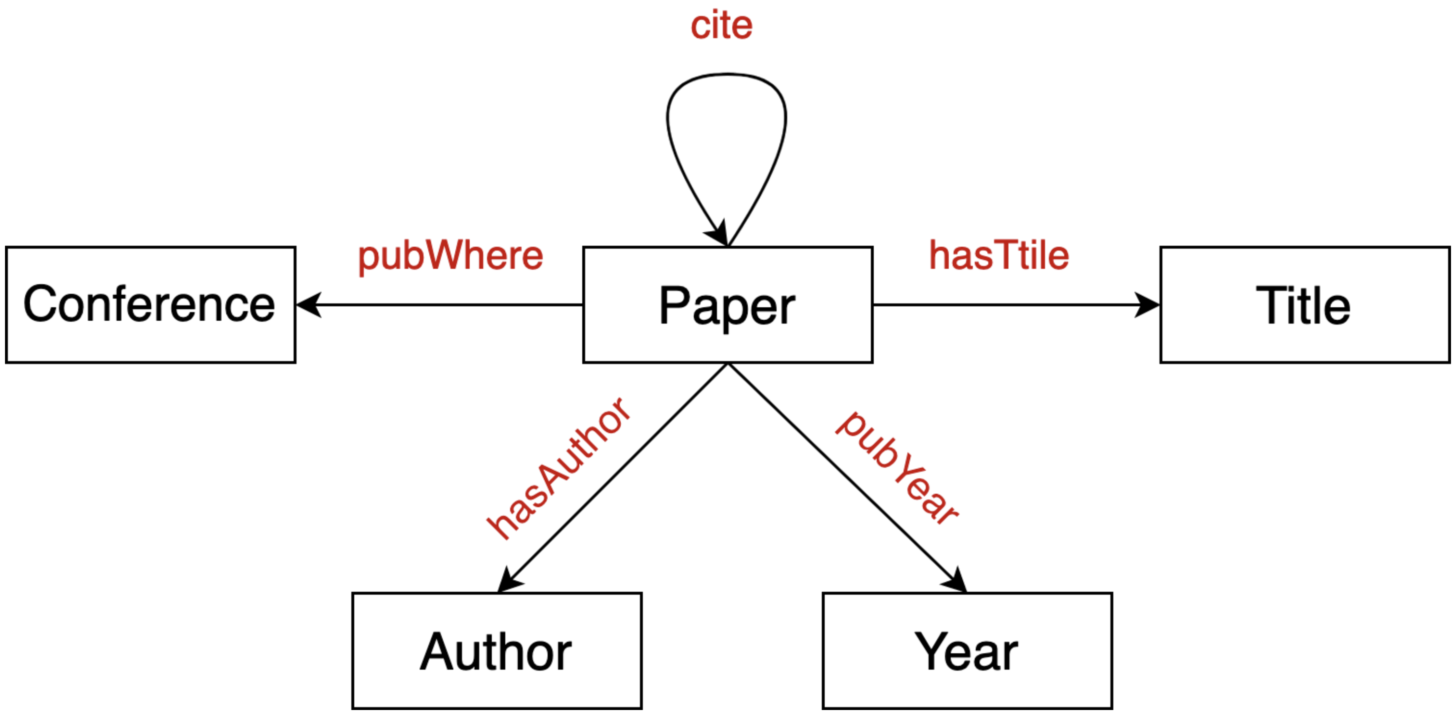
Knowledge in graph form:

- Capture entities, types, and relationships
- Nodes are entities
- Nodes are labeled with their types
- Edges between two nodes capture relationships between entities

- KG is an example of a heterogeneous graph

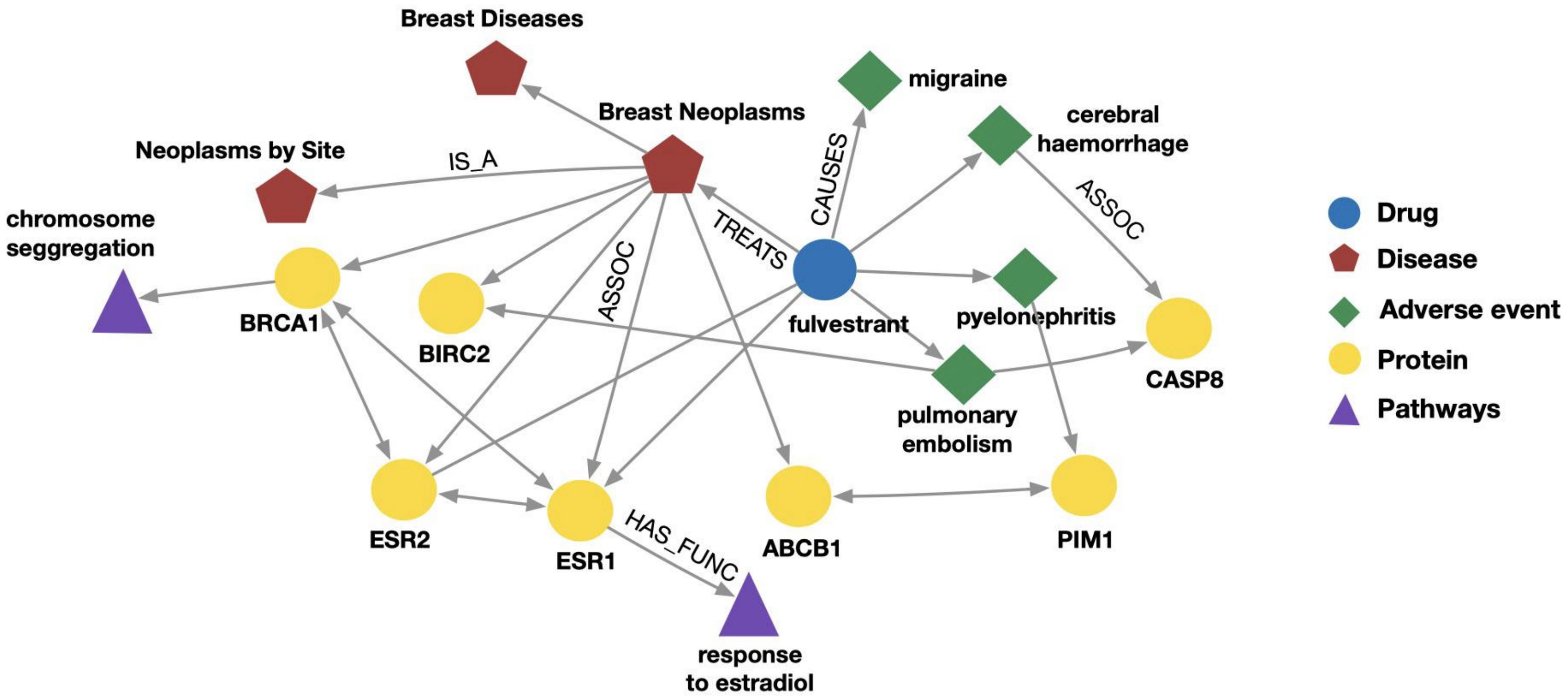
Example: Bibliographic Networks

- Node types: paper, title, author, conference, year
- Relation types: pubWhere, pubYear, hasTitle, hasAuthor, cite



Example: Bio Knowledge Graphs

- Node types: drug, disease, adverse event, protein, pathways
- Relation types: has_func, causes, assoc, treats, is_a



Knowledge Graph in Practice

Examples of knowledge graphs

- Google Knowledge Graph
- Amazon Product Graph
- Facebook Graph API
- IBM Watson
- Microsoft Satori
- Project Hanover/Literome
- LinkedIn Knowledge Graph
- Yandex Object Answer

Applications of Knowledge Graphs

- Serving information

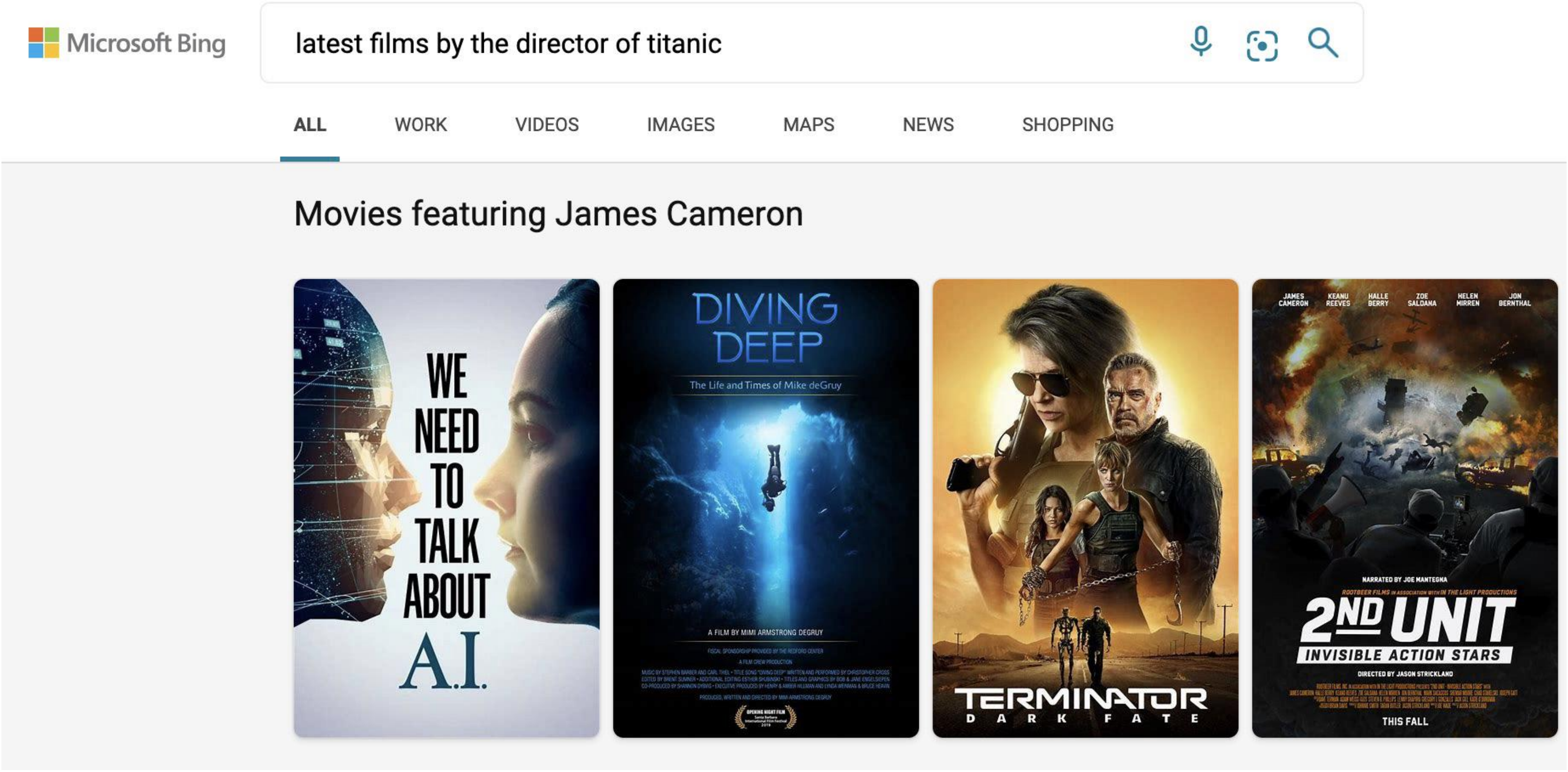


Image credit: Bing

Applications of Knowledge Graphs

- Question answering and conversation agents

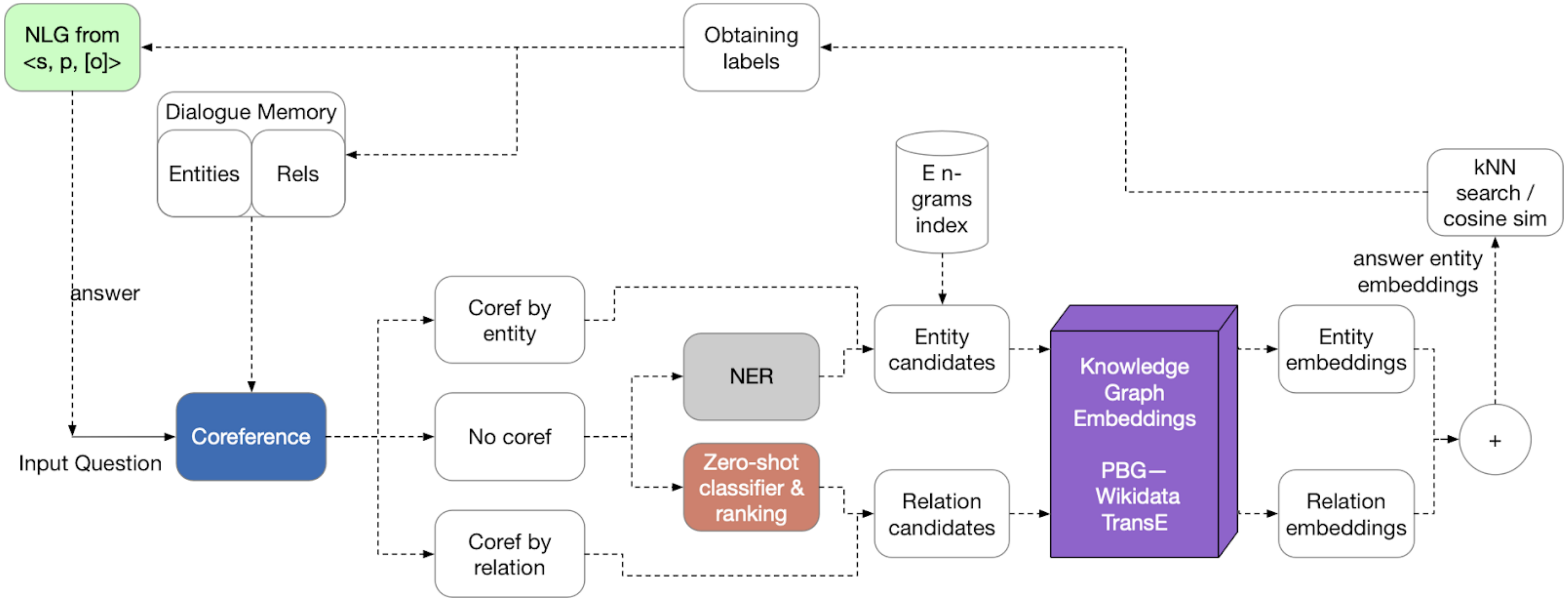


Image credit: [Medium]

Knowledge Graph Datasets

Publicly available KGs:

- FreeBase, Wikidata, Dbpedia, YAGO, NELL, etc.
- Common characteristics:
- Massive: Millions of nodes and edges
- Incomplete: Many true edges are missing

Given a massive KG, enumerating all the possible facts is intractable!

⇨

Can we predict plausible BUT missing links?

Example: Freebase

- Freebase
  - ~80 million entities
  - ~38K relation types
  - ~3 billion facts/triples
  - 93.8% of persons from Freebase have no place of birth and 78.5% have no nationality!

Freebase™
A socially managed semantic database

- Datasets: FB15k/FB15k-237
  - A complete subset of Freebase, used by researchers to learn KG models

Dataset	Entities	Relations	Training Edges	Validation Edges	Test Edges	Total Edges
FB15k	14,951	1,345	483,142	50,000	59,071	592,213
FB15k-237	14,505	237	272,115	17,526	20,438	310,079

[1] Paulheim, Heiko. "Knowledge graph refinement: A survey of approaches and evaluation methods." Semantic web 8.3 (2017): 489-508.
[2] Min, Bonan, et al. "Distant supervision for relation extraction with an incomplete knowledge base." Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2013.

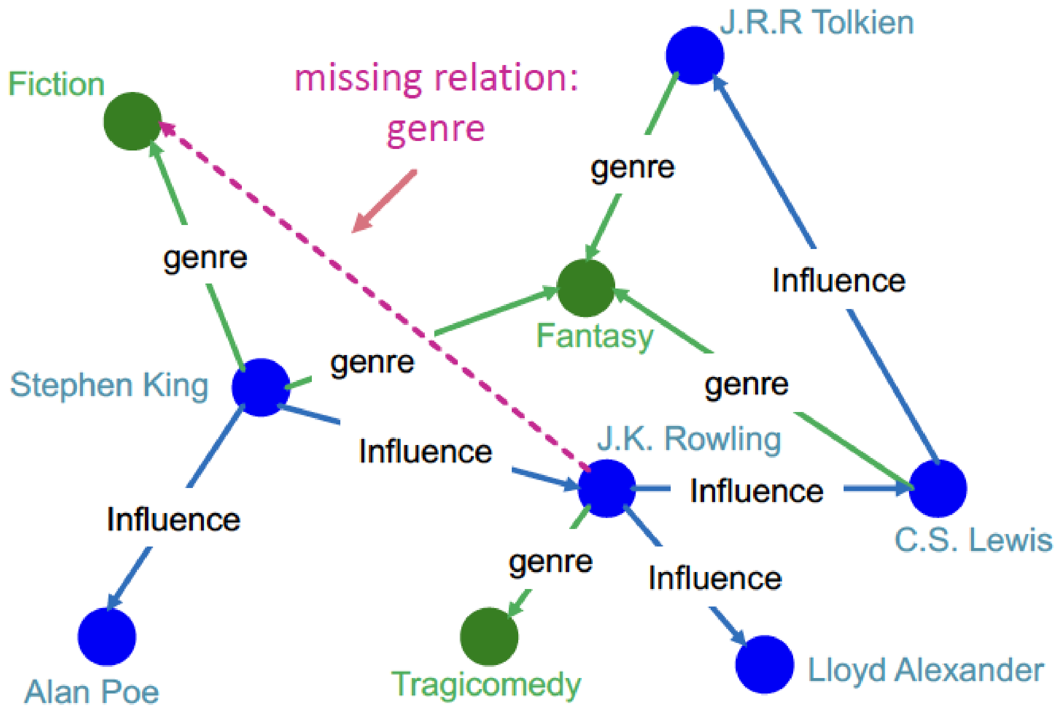
Knowledge Graph Completion

KG Completion Task

Given an enormous KG, can we complete the KG?

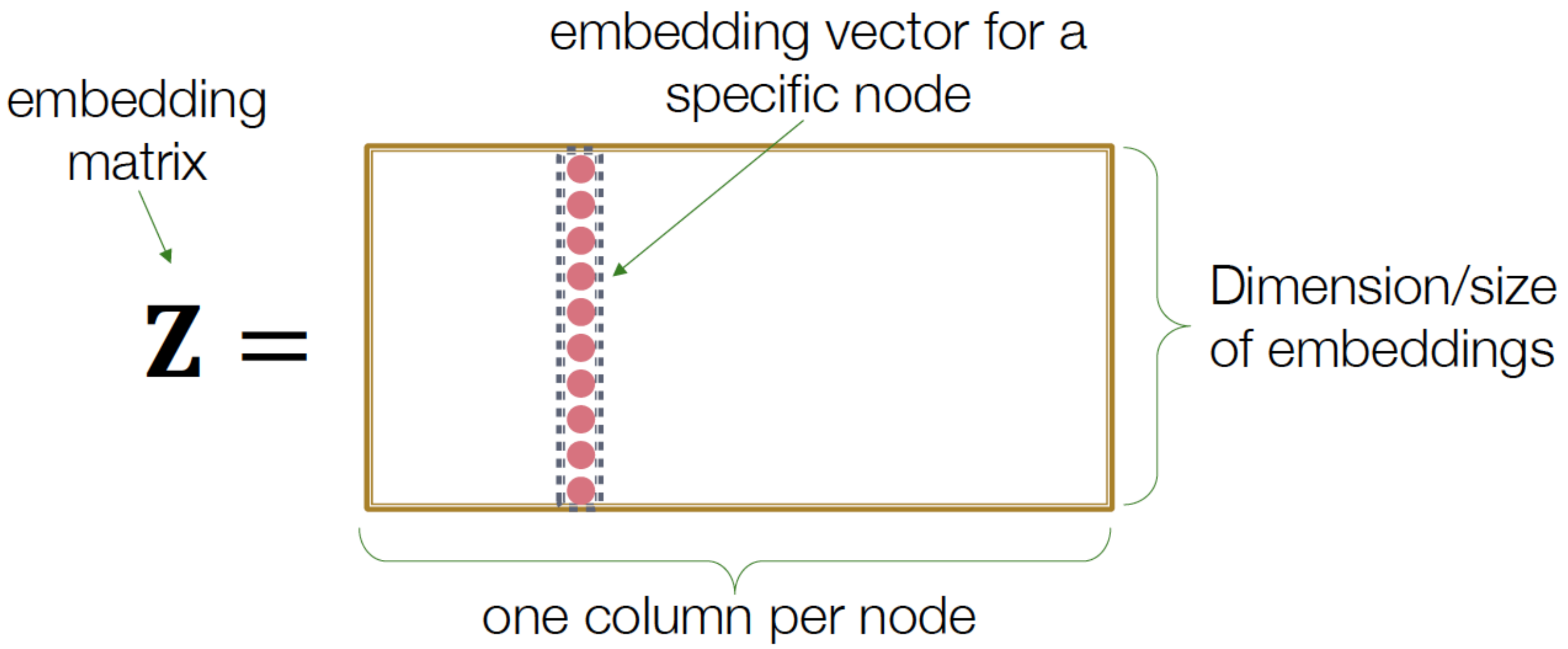
- For a given (head, relation), we predict missing tails.
- (Note this is slightly different from link prediction task)

Example task: predict the tail "Science Fiction" for ("J.K. Rowling", "genre")



Recap: Shallow Encoding

- Simplest encoding approach: Encoder is an embedding lookup

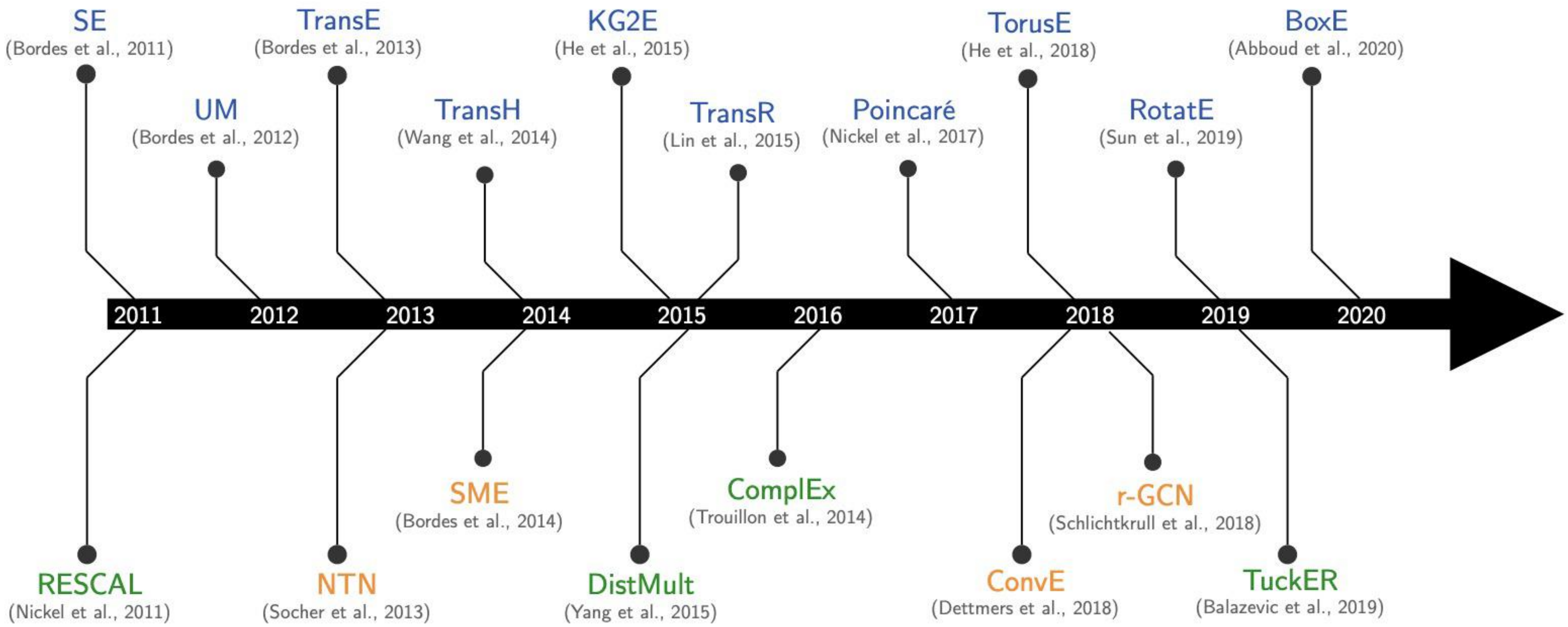


KG Representation

- Edges in KG are represented as triples (h, r, t)
  - head (h) has relation r with tail (t)

- Key Idea:
  - Model entities and relations in embedding space $\mathbb{R}^d$
  - Associate entities and relations with shallow embeddings
    - Note we do not learn a GNN here!
  - Given a triple (h, r, t), the goal is that the embedding of (h, r) should be close to the embedding of t.
    - How to embed (h, r)
    - How to define score $f_r(h, t)$?
      - Score f_r is high if h, r, t exists, else f_r is low

Many KG Embedding Models



Today: Different Models

- We are going to learn about different KG embedding models (shallow/transductive embs):
- Different models are…
  - …based on different geometric intuitions
  - …capture different types of relations (have different expressivity)

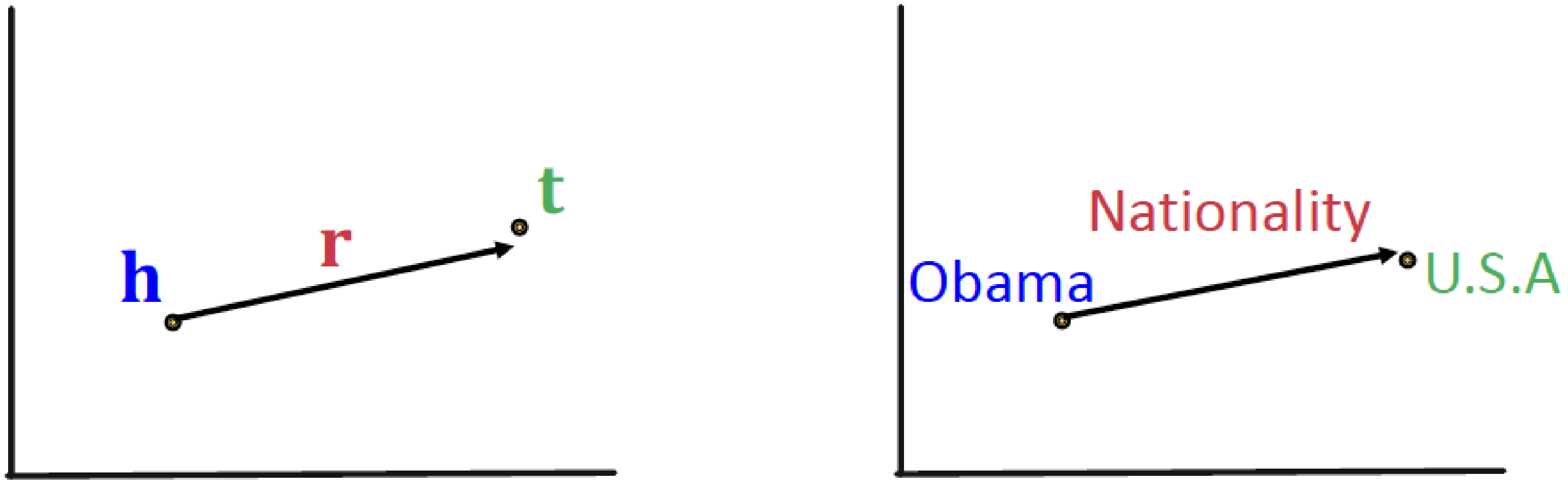
Model	Score	Embedding	Sym.	Antisym.	Inv.	Compos.	1-to-N
TransE	$-\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$	$\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{R}^k$	✗	✓	✓	✓	✗
TransR	$-\|\boldsymbol{M}_r\mathbf{h}+\mathbf{r}-\boldsymbol{M}_r\mathbf{t}\|$	$\mathbf{h},\mathbf{t}\in\mathbb{R}^k$, $\mathbf{r}\in\mathbb{R}^d$, $\boldsymbol{M}_r\in\mathbb{R}^{d\times k}$	✓	✓	✓	✓	✓
DistMult	$<\mathbf{h},\mathbf{r},\mathbf{t}>$	$\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{R}^k$	✓	✗	✗	✗	✓
ComplEx	$\mathrm{Re}(<\mathbf{h},\mathbf{r},\bar{\mathbf{t}}>)$	$\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{C}^k$	✓	✓	✓	✗	✓

Knowledge Graph Completion: TransE

Bordes et al., Translating embeddings for modeling multi-relational data, NeurIPS 2013.

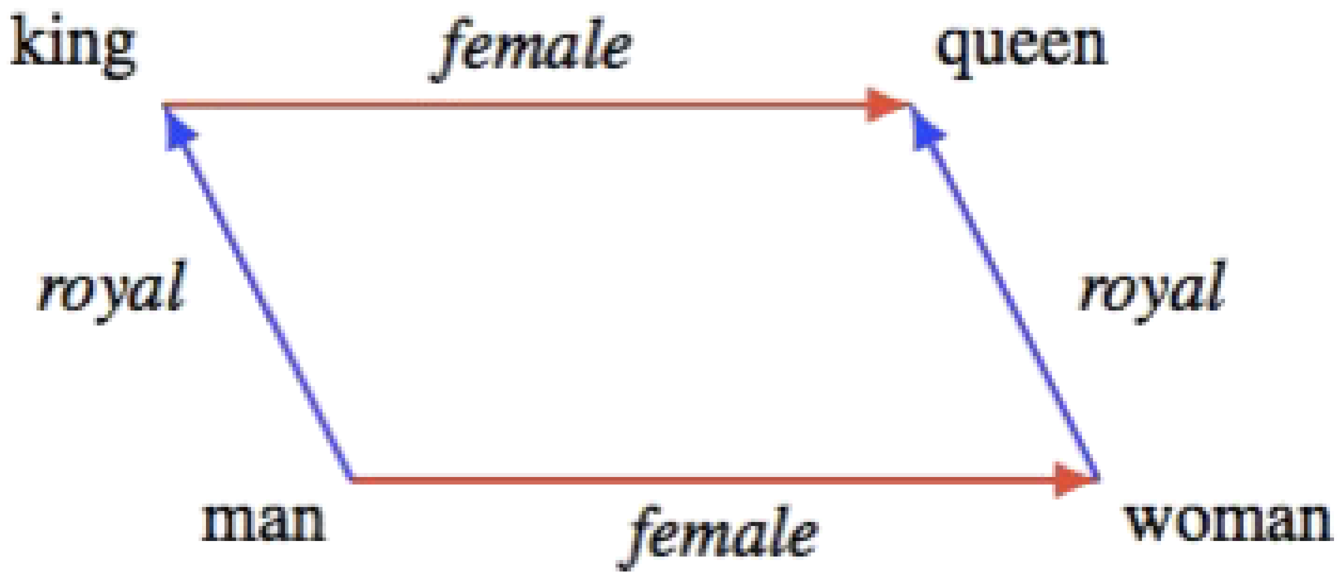
TransE

- Intuition: Translation
  - For a triple (h, r, t), let $h, r, t \in \mathbb{R}^d$ be embedding vectors
  - TransE: $h + r \approx t$ if the given link exists; else $h + r \neq t$
- Entity scoring function: $f_r(h, t) = -\|h + r - t\|$



TransE: Idea

- Entity embedding



TransE: How to Learn

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

end loop

Initialize relations r and entities e uniformly, then normalize.
γ is the margin.

Sample triplet (h', r, t) that does not appear in the KG.

d represents distance (negative of score)

Contrastive loss: Favors lower distance (or higher score) for valid triplets, high distance (or lower score) for corrupted ones

Connectivity Patterns in KG

- Relations in a heterogeneous KG have different properties:
  - Example:
    - **Symmetry**: If the edge (h, "Roommate", t) exists in KG, then the edge (t, "Roommate", h) should also exist.
    - **Inverse relation**: If the edge (h, "Advisor", t) exists in KG, then the edge t, "Advisee", h should also exist.
- Can we categorize these relation patterns?
- Are KG embedding methods (e.g., TransE) expressive enough to model these patterns?

Four Relation Patterns

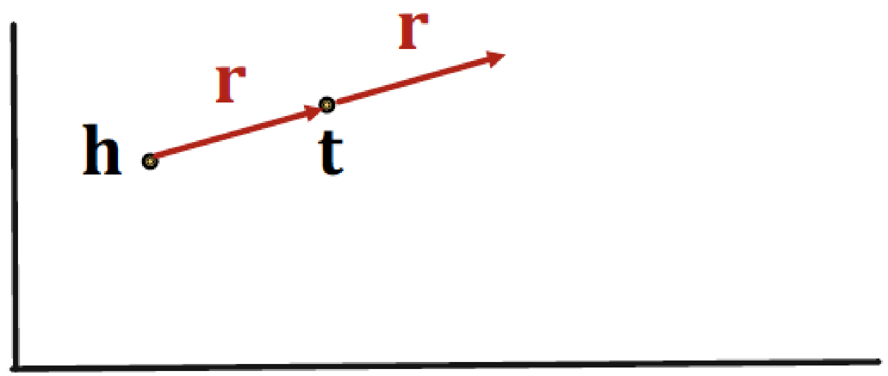
- Symmetric (Antisymmetric) Relations:

$$r(h,t) \Longrightarrow r(t,h) \quad \big(r(h,t) \Rightarrow \neg r(t,h)\big) \; \forall h,t$$

  - Example:
    - Symmetric: Family, Roommate
    - Antisymmetric: Hypernym (a word with a broader meaning: poodle vs. dog)
- Inverse Relations: $r_2(h,t) \Longrightarrow r_1(t,h)$
  - Example : (Advisor, Advisee)
- Composition (Transitive) Relations: $r_1(x,y) \wedge r_2(y,z) \Longrightarrow r_3(x,z), \forall x,y,z$
  - Example: My mother's husband is my father.
- 1-to-N relations: $r_1(h,t_1), r_1(h,t_2), \cdots, r(h,t_n)$ are all True
  - Example: r is "StudentsOf"

Antisymmetric Relation in TransE

- Antisymmetric Relations: $r(h,t) \Rightarrow \neg r(t,h)$
  - Example: Hypernym (a word with a broader meaning: poodle vs. dog)
- TransE can model antisymmetric relations
  - $h + r = t$ but $\mathrm{t} + r \neq h$

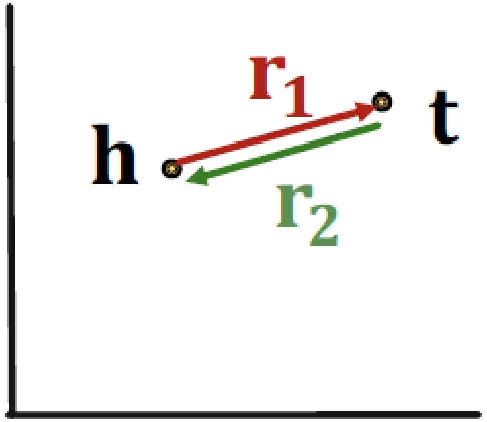


Inverse Relations in TransE

- Inverse Relations:

$$r_2(h,t) \Rightarrow r_1(t,h)$$

  - Example : (Advisor, Advisee)
- TransE can model inverse relations
  - $h + r_2 = t$, we can set $r_1 = -r_2$



Composition in TransE

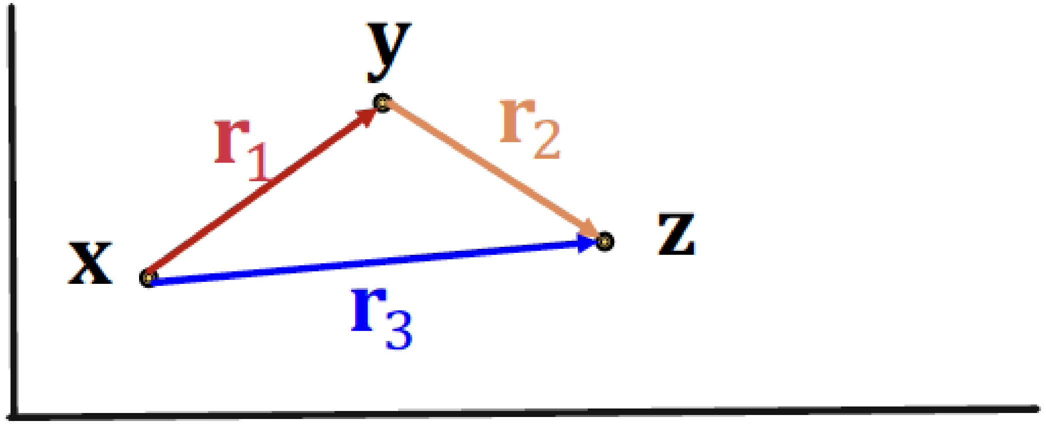
- Composition (Transitive) Relations:

$$r_1(x, y) \wedge r_2(y, z) \Longrightarrow r_3(x, z), \forall x, y, z$$

  - Example: My mother's husband is my father.

- TransE can model composition relations

$$r_3 = r_1 + r_2$$

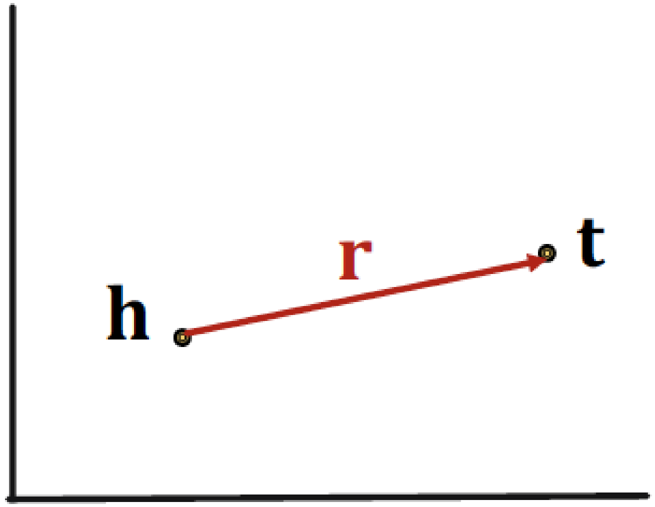


Limitation: Symmetric Relations

- Symmetric Relations:

$$r(h,t) \Rightarrow r(t,h) \quad \forall h,t$$

  - Example: Family, Roommate
- TransE cannot model symmetric relations
  - Only if $r = 0, h = t$

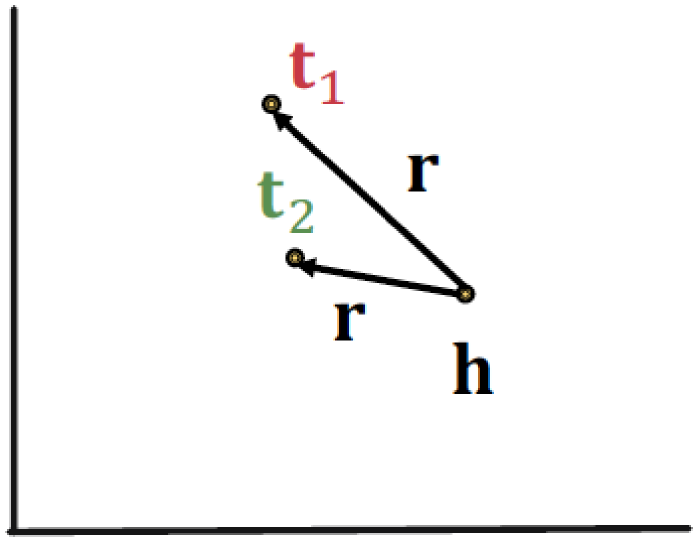


For all h, t that satisfy $\mathrm{r(h,t)}$, $\mathrm{r(t,h)}$ is also True, which means **h** + **r** − **t** = 0 and **t** + **r** − **h** = 0. Then **r** = 0 and **h** = **t**. However, *h* and *t* are two different entities and should be mapped to different locations.

Limitation: 1-to-N Relation

- 1-to-N Relations:
  - Example: (h, r, t_1) and (h, r, t_2) both exist in the knowledge graph, e.g., r is "StudentsOf"
- TransE cannot model 1-to-N relations
  - t_1 and t_2 will map to the same vector, although they are different entities
- $t_1 = h + r = t_2$
- $t_1 \neq t_2$

contradictory!



Today: KG Completion Models

- What we learned so far:

Model	Score	Embedding	Sym.	Antisym.	Inv.	Compos.	1-to-N
TransE	$-\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$	$\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{R}^k$	✗	✓	✓	✓	✗

Knowledge Graph Completion: TransR

Lin et al., Learning entity and relation embeddings for knowledge graph completion, AAAI 2015

TransR

- TransE models translation of any relation in the same embedding space.

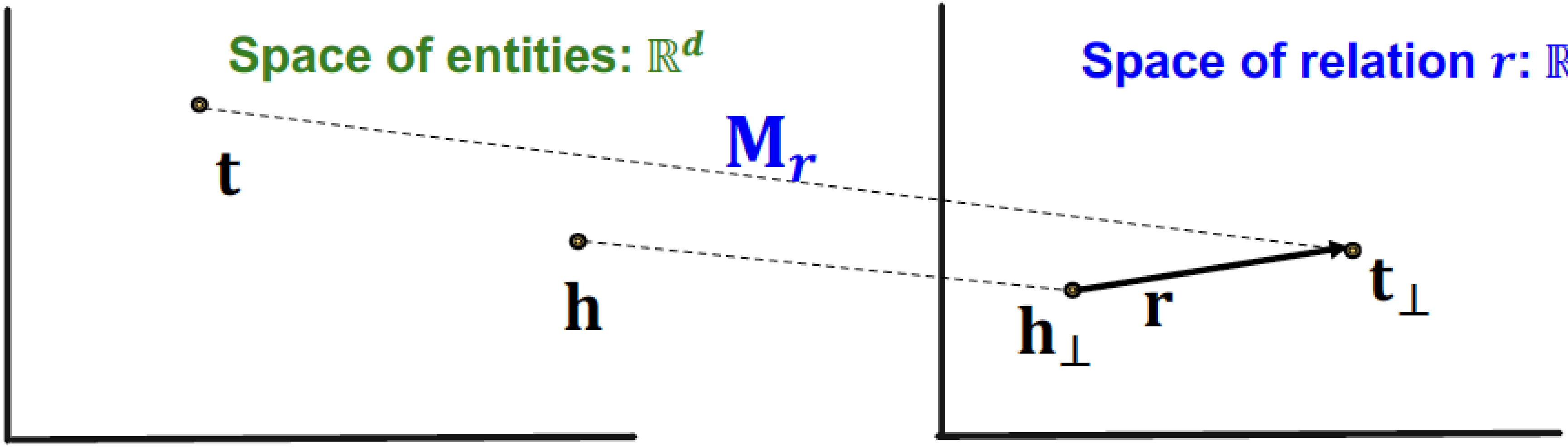
- Can we design a new space for each relation and do translation in relation-specific space?

- TransR: model entities as vectors in the entity space $\mathbb{R}^d$ and model each relation as vector in relation space $\mathbf{r} \in \mathbb{R}^k$ with $\mathbf{M}_r \in \mathbb{R}^{k*d}$ as the projection matrix.

TransR

- TransR: model entities as vectors in the entity space $\mathbb{R}^{d}$ and model each relation as vector in relation space $r \in \mathbb{R}^{k}$ with $M_r \in \mathbb{R}^{k \times d}$ as the projection matrix.
- $h_\perp = M_r h, t_\perp = M_r t$
- Score function: $f_r(h,t) = -\|h_\perp + r - t_\perp\|$

Use M_r to project from entity space $\mathbb{R}^d$ to relation space $\mathbb{R}^k$!!



Symmetric Relations in TransR

- Symmetric relations:

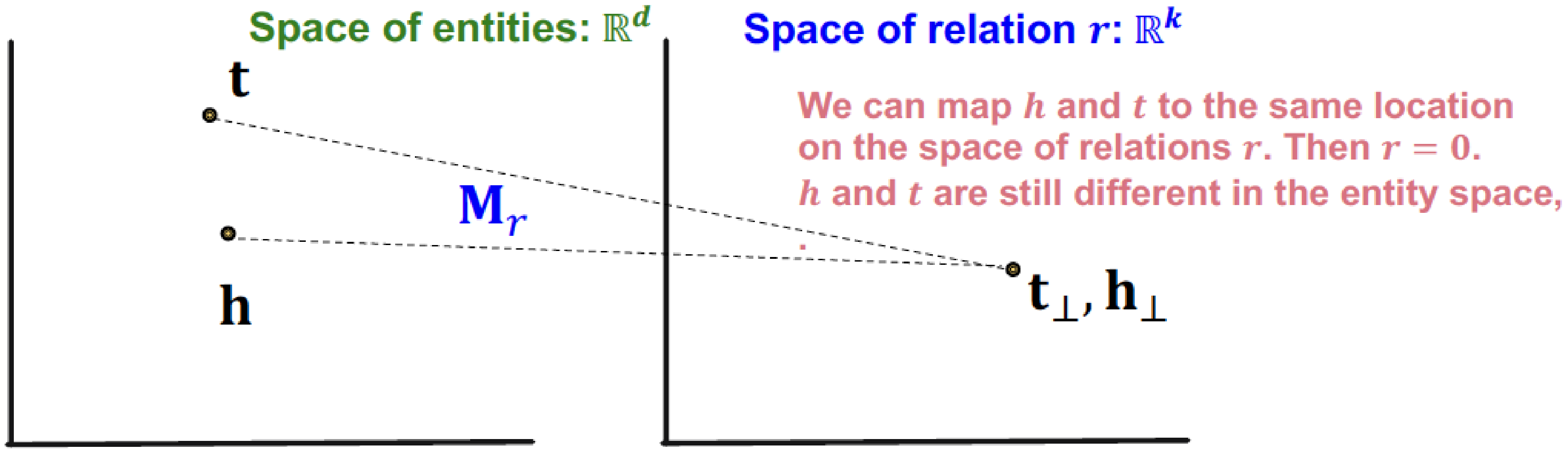
$$r(h,t) \Rightarrow r(t,h) \quad \forall h,t$$

  - Example: Family, Roommate

Note different symmetric relations may have different M_r

- TransR can model symmetric relations

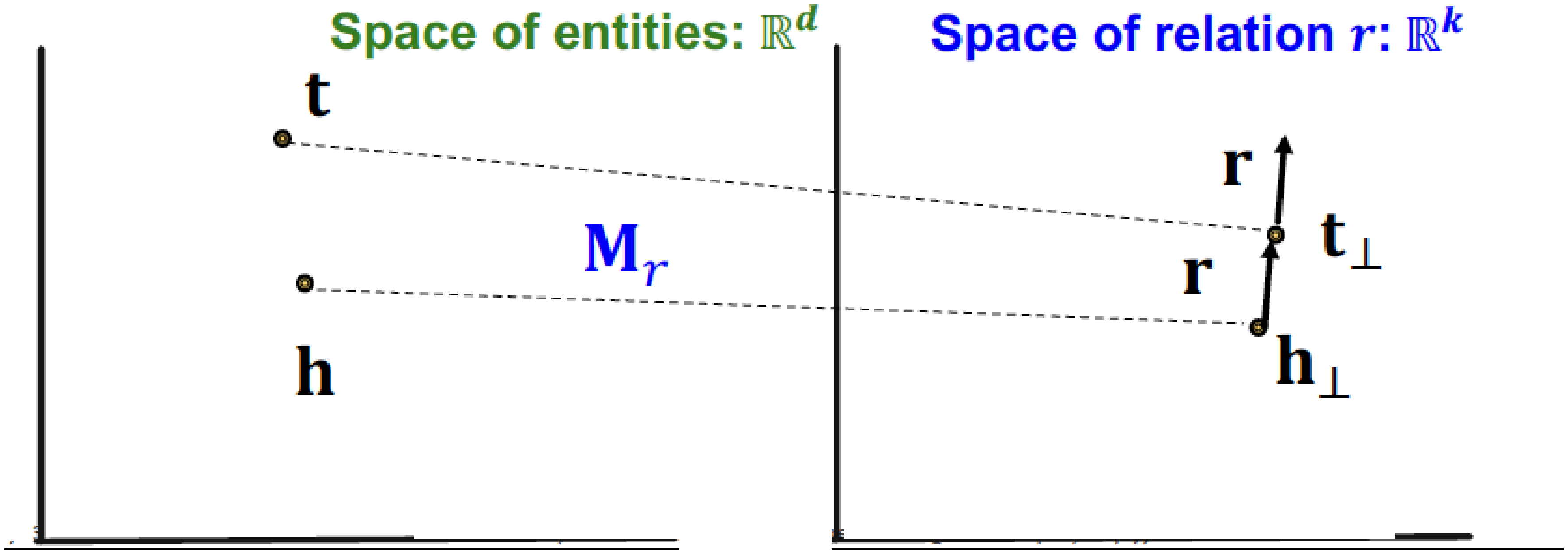
$$r = 0, h_\perp = M_r h = M_r t = t_\perp$$



Antisymmetric Relation in TransR

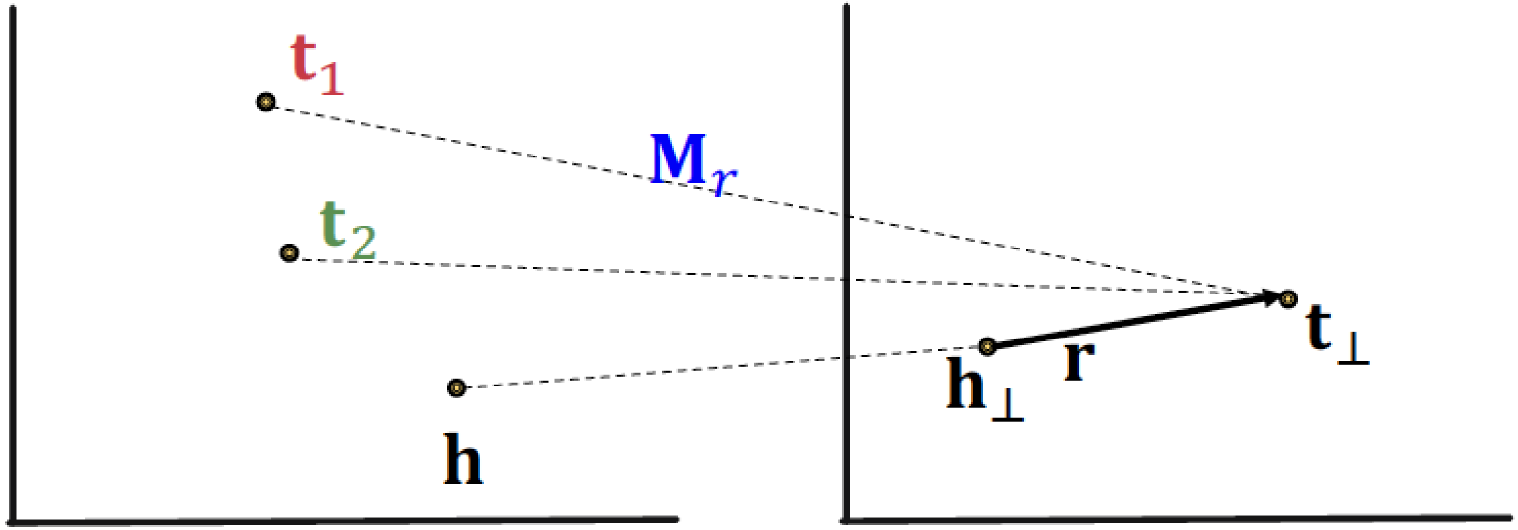
- Antisymmetric Relations: $r(h, t) \Rightarrow \neg r(t, h)$
  - Example: Hypernym (a word with a broader meaning: poodle vs. dog)
- TransR can model antisymmetric relations

$$r \neq 0, M_r h + r = M_r t \Longrightarrow M_r t + r \neq M_r h$$



1-to-N Relations in TransR

- 1-to-N Relations:
  - Example: (h, r, t_1) and (h, r, t_2) both exist in the knowledge graph, e.g., r is "StudentsOf"
- TransR can model 1-to-N relations
  - We can learn M_r so that $t_\perp = \mathrm{M_r t_1} = \mathrm{M_r t_2}$
  - Note that $\mathrm{t_1}$ does not need to be equal to $\mathrm{t_2}$



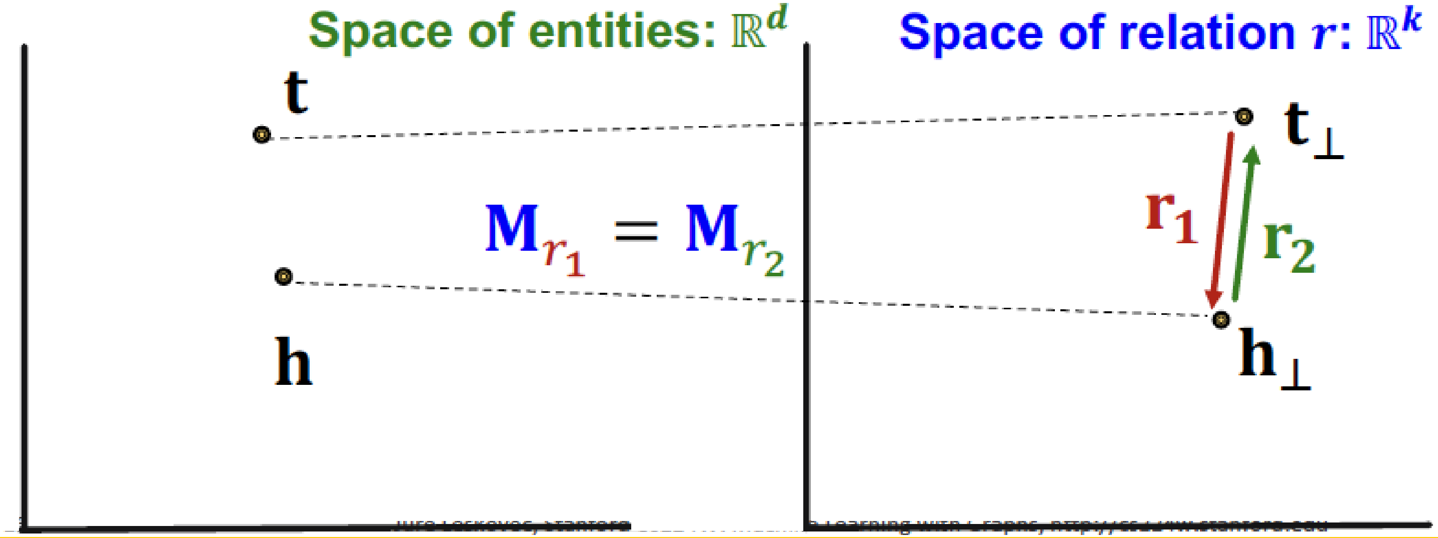
Inverse Relations in TransR

- Inverse Relations:

$$r_2(h,t) \Rightarrow r_1(t,h)$$

  - Example : (Advisor, Advisee)
- TransR can model inverse relations

$$r_2 = -r_1, M_{r_1} = M_{r_2} \Longrightarrow M_{r_1}t + r_1 = M_{r_1}h \text{ and } M_{r_2}h + r_2 = M_{r_2}r$$



Composition Relation in TransR

- Composition (Transitive) Relations:

$$r_1(x, y) \wedge r_2(y, z) \Longrightarrow r_3(x, z), \forall x, y, z$$

  - Example: My mother's husband is my father.

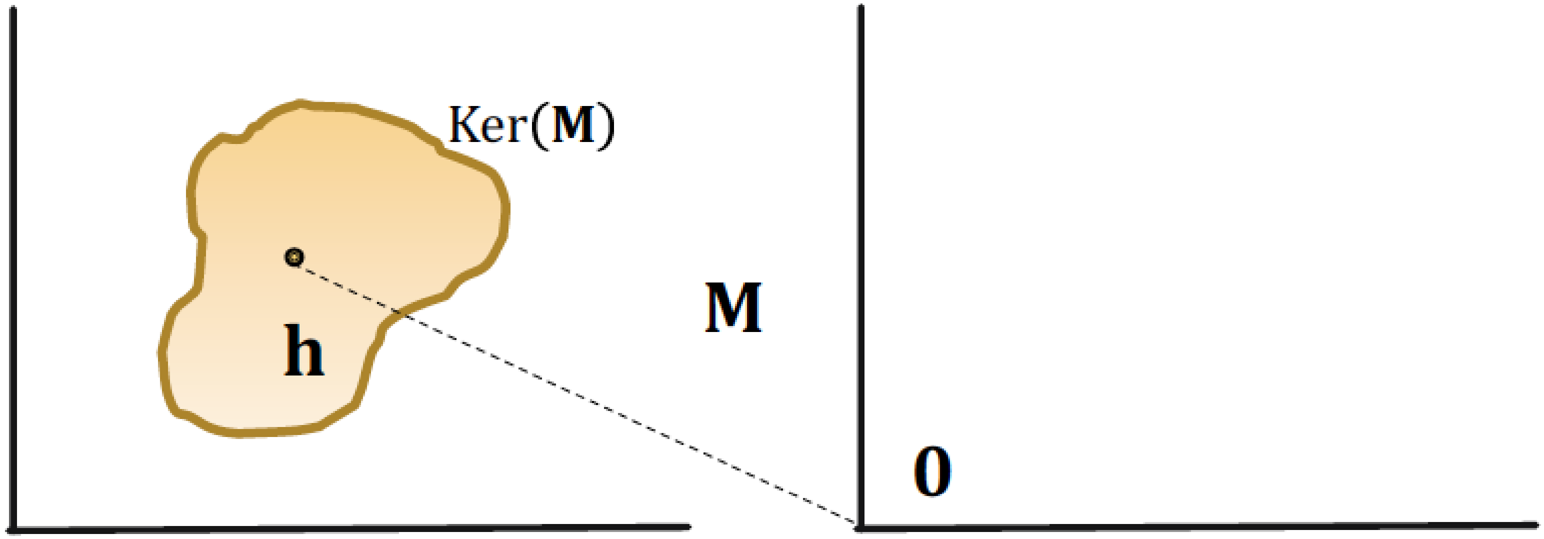
- TransR can model composition relations
  - High-level intuition: TransR models a triple with linear functions. Linear functions are chainable!
    - If f(x) and g(x) are linear, then f(g(x)) is also linear
    - Let: f(x)=ax+b, g(x)=cx+d: then f(g(x))= a(cx+d)+b.

Composition Relation in TransR

- Composition (Transitive) Relations:

$$r_1(x, y) \wedge r_2(y, z) \Longrightarrow r_3(x, z), \forall x, y, z$$

  - Example: My mother's husband is my father.
- Background:
  - Def: Kernel space of a matrix **M**: $h \in Kernel(M), then\ M \cdot h = 0$



Composition Relation in TransR

- Composition Relations:

$$r_1(x, y) \wedge r_2(y, z) \Longrightarrow r_3(x, z), \forall x, y, z$$

- Assume $M_{r_1} g_1 = r_1$ and $M_{r_2} g_2 = r_2$
- For $r_1(x, y)$:
  - $r_1(x, y)$ exists $\Longrightarrow M_{r_1} x + r_1 = M_{r_1} y \Longrightarrow M_{r_1}(y - x) = r_1$
  - $\Longrightarrow y - x - g_1 \in Ker\left(M_{r_1}\right) \Longrightarrow y \in x + g_1 + Ker\left(M_{r_1}\right)$
- For $r_2(y, z)$
  - $r_2(y, z)$ exists $\Longrightarrow M_{r_2} y + r_2 = M_{r_2} z \Longrightarrow M_{r_2}(z - y) = r_2$
  - $\Longrightarrow z - y - g_2 \in Ker\left(M_{r_2}\right) \Longrightarrow z \in y + g_2 + Ker\left(M_{r_2}\right)$
- Then we have:

$$z \in x + g_1 + g_2 + Ker\left(M_{r_1}\right) + Ker\left(M_{r_2}\right)$$

Composition Relation in TransR

- Composition Relations:

$$r_1(x, y) \wedge r_2(y, z) \Longrightarrow r_3(x, z), \forall x, y, z$$

We have: $z \in x + g_1 + \ g_2 + Ker\left(M_{r_1}\right) + Ker\left(M_{r_2}\right)$

- Construct M_{r_3} such that: $Ker\left(M_{r_3}\right) = Ker\left(M_{r_1}\right) + Ker\left(M_{r_2}\right)$
  - M_{r_3} always exists

- Set $r_3 = M_{r_3}(g_1 + g_2)$
- We have: $M_{r_3}x + r_3 = M_{r_3}z$

Today: KG Completion Models

- What we have learnt so far:

Model	Score	Embedding	Sym.	Antisym.	Inv.	Compos.	1-to-N
TransE	$-\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$	$\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{R}^k$	✗	✓	✓	✓	✗
TransR	$-\|\boldsymbol{M}_r\mathbf{h}+\mathbf{r}-\boldsymbol{M}_r\mathbf{t}\|$	$\mathbf{h},\mathbf{t}\in\mathbb{R}^k$, $\mathbf{r}\in\mathbb{R}^d$, $\boldsymbol{M}_r\in\mathbb{R}^{d\times k}$	✓	✓	✓	✓	✓

Knowledge Graph Completion: DistMult

Yang et al, Embedding Entities and Relations for Learning and Inference in Knowledge Bases, ICLR 2015

New Idea: Bilinear Modeling

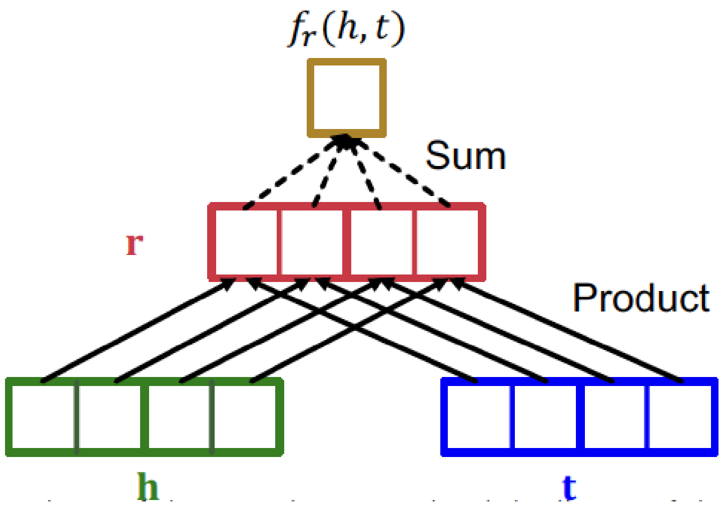
- So far: The scoring function $f_r(h, t)$ is negative of L1 / L2 distance in TransE and TransR.
- Idea: Use bilinear modeling:
  - Score function: $f_r(h, t) = h \cdot A \cdot t$, where $h, t \in \mathbb{R}^k, A \in \mathbb{R}^{k \times k}$
- Problem: Too general and prone to overfitting
  - Matrix A is too expressive
- Fix: Limit A to be diagonal
  - This is called DistMult

New Idea: Bilinear Modeling

- DistMult: Entities & relations are vectors in $\mathbb{R}^k$
- Score function:

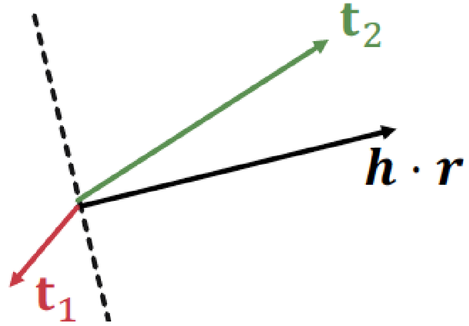
$$h_r(h,t) = \langle h, r, t \rangle = \sum_i h_i \cdot r_i \cdot t_i$$

- $h, r, t \in \mathbb{R}^k$



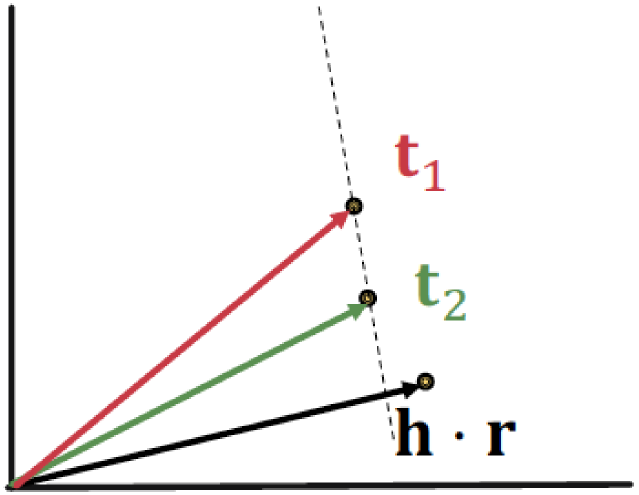
DistMult

- DistMult: Entities and relations using vectors in $\mathbb{R}^k$
- Score function: $f_r(h,t) = \langle h, r, t \rangle = \sum_i h_i \cdot r_i \cdot t_i$
- $h, r, t \in \mathbb{R}^k$
- Intuition of the score function: Can be viewed as a cosine similarity between $h \odot r$ and t where $\odot$ is the Hadamard product (elementwise product)
- Example: $f_r(h,t_1) < 0, \quad f_r(h,t_2) > 0$



1-to-N Relations in DistMult

- 1-to-N Relations:
  - Example: (h, r, t_1) and (h, r, t_2) both exist in the knowledge graph, e.g., r is "StudentsOf"
- DistMult can model 1-to-N relations $\langle h, r, t_1 \rangle = \langle h, r, t_2 \rangle$



Symmetric Relations in DistMult

- Symmetric Relations:

$$r(h,t) \Longrightarrow r(t,h) \quad \forall h,t$$

  - Example: Family, Roommate
- DistMult can model symmetric relations

$$f_r(h,t) = \langle h, r, t \rangle = \sum_i h_i \cdot r_i \cdot t_i$$

$$= \sum_i t_i \cdot r_i \cdot h_i = \langle t, r, h \rangle = f_r(t,h)$$

Due to the commutative property of multiplication.

Limitation: Antisymmetric Relations

- Antisymmetric Relations: $r(h,t) \Rightarrow \neg r(t,h)$
  - Example: Hypernym (a word with a broader meaning: poodle vs. dog)
- DistMult cannot model antisymmetric relations
  - $f_r(h,t) = f_r(t,h)$: $r(h,t)$ and $r(t,h)$ always have same score!

DistMult cannot differentiate between head entity and tail entity! This means that all relations are modelled as symmetric regardless, i.e., even anti-symmetric relations will be represented as symmetric.

Limitation: Inverse Relations

- Inverse Relations:

$$r_2(h,t) \Rightarrow r_1(t,h)$$

  - Example : (Advisor, Advisee)

- DistMult cannot model inverse relations
  - Assume DistMult does model inverse relations:
  - $f_{r_2}(h,t) = \langle h, r_2, t \rangle = \langle t, r_1, h \rangle = f_{r_1}(t,h)$
  - For example, $r_2 = r_1$ solves this (there also exist solutions $r_2 \neq r_1$)
  - But semantically this does not make sense: The embedding of "Advisor" relation should not be the same as "Advisee" relation.

Limitation: Composition Relations

- Composition (Transitive) Relations:

$$r_1(x, y) \wedge r_2(y, z) \Longrightarrow r_3(x, z), \forall x, y, z$$

  - Example: My mother's husband is my father.

- DistMult cannot model composition of relations
- Intuition: Because dot product is commutative ($a \cdot b = \mathrm{b} \cdot a$), DistMult does not distinguish between head and tail entities, so it cannot model composition.

Today: KG Completion Models

- What we learned so far:

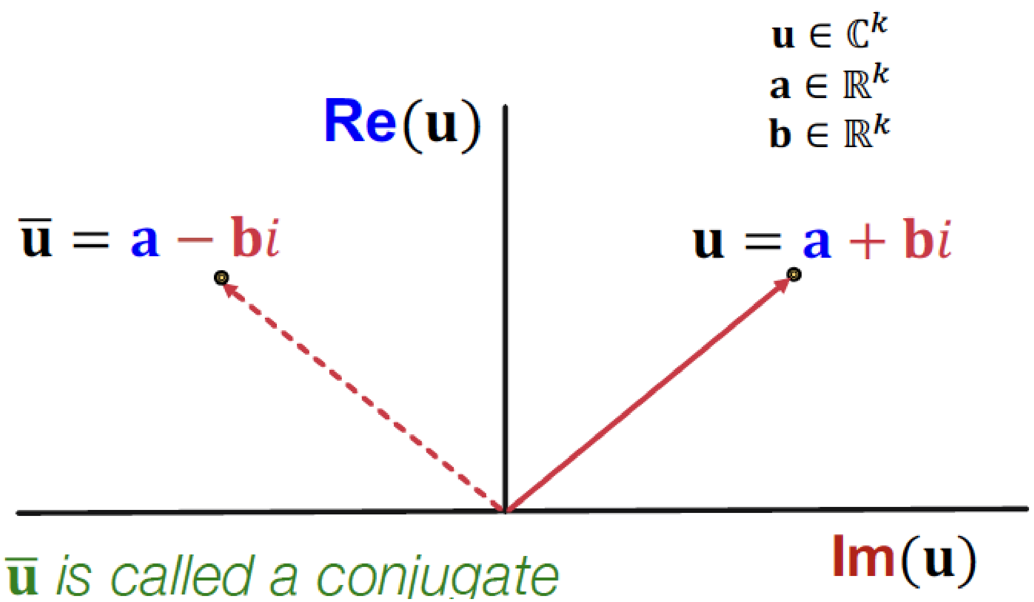
Model	Score	Embedding	Sym.	Antisym.	Inv.	Compos.	1-to-N
TransE	$-\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$	$\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{R}^k$	✗	✓	✓	✓	✗
TransR	$-\|\boldsymbol{M}_r\mathbf{h}+\mathbf{r}-\boldsymbol{M}_r\mathbf{t}\|$	$\mathbf{h},\mathbf{t}\in\mathbb{R}^k$, $\mathbf{r}\in\mathbb{R}^d$, $\boldsymbol{M}_r\in\mathbb{R}^{d\times k}$	✓	✓	✓	✓	✓
DistMult	$<\mathbf{h},\mathbf{r},\mathbf{t}>$	$\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{R}^k$	✓	✗	✗	✗	✓

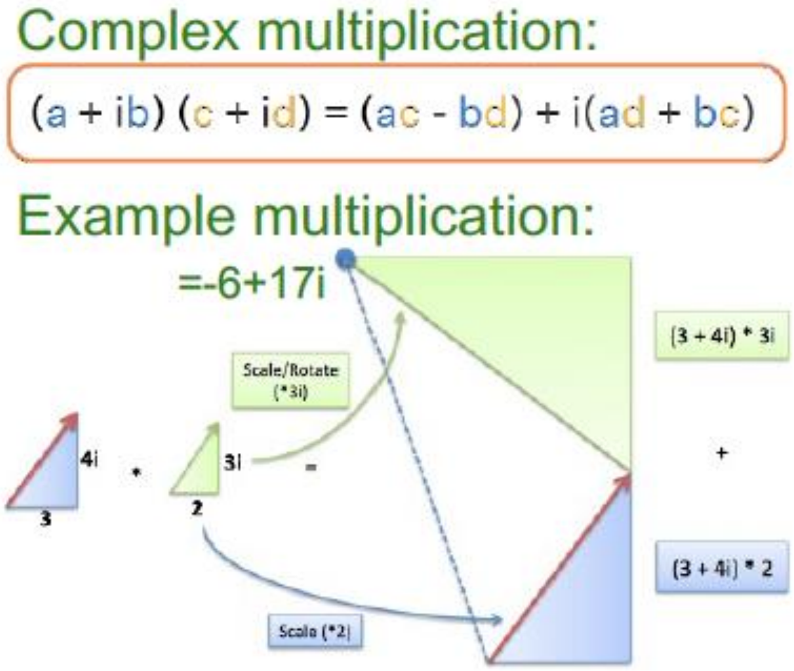
Knowledge Graph Completion: ComplEx

Trouillon et al, *Complex Embeddings for Simple Link Prediction*, ICML 2016

ComplEx

- Based on Distmult, ComplEx embeds entities and relations in Complex vector space
- ComplEx: model entities and relations using vectors in $\mathbb{C}^k$

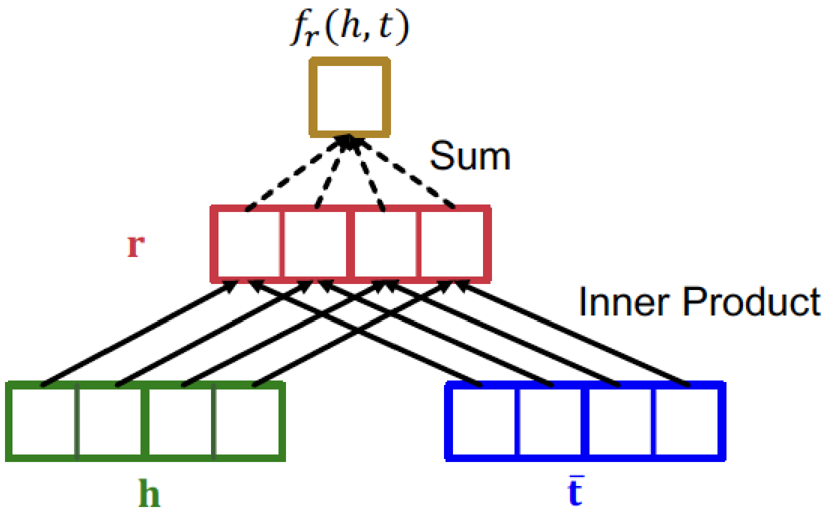




ComplEx

- Based on Distmult, ComplEx embeds entities and relations in Complex vector space
- ComplEx: model entities and relations using vectors in $\mathbb{C}^k$
- Score function: $f_r(h,t) = \mathrm{Re}(\sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{t}}_i)$

$$
\begin{aligned}
&= \langle \mathrm{Re}(\mathbf{h}_i), \mathrm{Re}(\mathbf{r}_i), \mathrm{Re}(\mathbf{t}_i) \rangle \\
&+ \langle \mathrm{Re}(\mathbf{h}_i), \mathrm{Im}(\mathbf{r}_i), \mathrm{Im}(\mathbf{t}_i) \rangle \\
&+ \langle \mathrm{Im}(\mathbf{h}_i), \mathrm{Re}(\mathbf{r}_i), \mathrm{Im}(\mathbf{t}_i) \rangle \\
&- \langle \mathrm{Im}(\mathbf{h}_i), \mathrm{Im}(\mathbf{r}_i), \mathrm{Re}(\mathbf{t}_i) \rangle
\end{aligned}
$$



ComplEx Score Function

$$f_r(h,t) = \mathrm{Re}\left(\sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{t}}_i\right)$$

$$= \sum_i \mathrm{Re}(\mathbf{h}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{t}}_i)$$

$$= \sum_i \mathrm{Re}((\mathrm{Re}(\mathbf{h}_i) + i\mathrm{Im}(\mathbf{h}_i)) \cdot (\mathrm{Re}(\mathbf{r}_i) + i\mathrm{Im}(\mathbf{r}_i)) \cdot (\mathrm{Re}(\mathbf{t}_i) - i\mathrm{Im}(\mathbf{t}_i)))$$

$$= \sum_i \begin{array}{l} \mathrm{Re}(\mathbf{h}_i)\mathrm{Re}(\mathbf{r}_i)\mathrm{Re}(\mathbf{t}_i) + \mathrm{Re}(\mathbf{h}_i)\mathrm{Im}(\mathbf{r}_i)\mathrm{Im}(\mathbf{t}_i) \\ +\mathrm{Im}(\mathbf{h}_i)\mathrm{Re}(\mathbf{r}_i)\mathrm{Im}(\mathbf{t}_i) - \mathrm{Im}(\mathbf{h}_i)\mathrm{Im}(\mathbf{r}_i)\mathrm{Re}(\mathbf{t}_i) \end{array}$$

$$= \langle \mathrm{Re}(\mathbf{h}_i), \mathrm{Re}(\mathbf{r}_i), \mathrm{Re}(\mathbf{t}_i) \rangle + \langle \mathrm{Re}(\mathbf{h}_i), \mathrm{Im}(\mathbf{r}_i), \mathrm{Im}(\mathbf{t}_i) \rangle$$
$$+ \langle \mathrm{Im}(\mathbf{h}_i), \mathrm{Re}(\mathbf{r}_i), \mathrm{Im}(\mathbf{t}_i) \rangle - \langle \mathrm{Im}(\mathbf{h}_i), \mathrm{Im}(\mathbf{r}_i), \mathrm{Re}(\mathbf{t}_i) \rangle$$

Antisymmetric Relations in ComplEx

- Antisymmetric Relations: $r(h,t) \Rightarrow \neg r(t,h)$
  - Example: Hypernym (a word with a broader meaning: poodle vs. dog)

- ComplEx can model antisymmetric relations
- The model is expressive enough to learn
  - $f_r(h,t) = \mathrm{Re}(\sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{t}}_i)$
  - $f_r(t,h) = \mathrm{Re}(\sum_i \boldsymbol{t}_i \cdot \mathbf{r}_i \cdot \bar{\boldsymbol{h}}_i)$
- Due to the asymmetric modeling using complex conjugate.

Symmetric Relations in ComplEx

- Symmetric Relations:

$$r(h,t) \Longrightarrow r(t,h) \quad \forall h,t$$

  - Example: Family, Roommate
- ComplEx can model symmetric relations
  - When $Im(r) = 0$, we have:

$$f_r(h,t) = \mathrm{Re}(\sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{t}}_i) = \sum_i \mathrm{Re}(\mathbf{r}_i \cdot \mathbf{h}_i \cdot \bar{\mathbf{t}}_i) = \sum_i \mathbf{r}_i \cdot \mathrm{Re}(\mathbf{h}_i \cdot \bar{\mathbf{t}}_i) = \sum_i \mathbf{r}_i \cdot \mathrm{Re}(\bar{\mathbf{h}}_i \cdot \mathbf{t}_i) = \sum_i \mathrm{Re}(\mathbf{r}_i \cdot \bar{\mathbf{h}}_i \cdot \mathbf{t}_i) = f_r(t,h)$$

Inverse Relations in ComplEx

- Inverse Relations:

$$r_2(h,t) \Rightarrow r_1(t,h)$$

  - Example : (Advisor, Advisee)

- ComplEx can model inverse relations
  - $r_1 = \bar{r}_2$
  - Complex conjugate of $r_2 = \underset{r}{\operatorname{argmax}}\, Re(\langle h, r, \bar{t} \rangle)$ is exactly $r_1 = \underset{r}{\operatorname{argmax}}\, Re\big(\langle t, r, \bar{h} \rangle\big)$

Composition and 1-to-N

- Composition (Transitive) Relations:

$$r_1(x,y) \wedge r_2(y,z) \Longrightarrow r_3(x,z), \forall x,y,z$$

  - Example: My mother's husband is my father.
- 1-to-N Relations:
  - Example: (h,r,t_1) and (h,r,t_2) both exist in the knowledge graph, e.g., r is "StudentsOf"

- ComplEx share the same property with DistMult
  - Cannot model composition relations
  - Can model 1-to-N relations

KG Embeddings in Practice

1. Different KGs may have drastically different relation patterns!

2. There is not a general embedding that works for all KGs, use the table to select models

3. Try TransE for a quick run if the target KG does not have much symmetric relations

4. Then use more expressive models, e.g., ComplEx, RotatE (TransE in Complex space)

Summary

- Link prediction / Graph completion is one of the prominent tasks on knowledge graphs
- Introduce TransE / TransR / DistMult / ComplEx models with different embedding space and expressiveness
- Next: Reasoning in Knowledge Graphs