Stanford CS224W: Heterogeneous Graphs and Knowledge Graph Embeddings

CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

Today: Heterogeneous Graphs

- **Goal**:
  - So far we only handle graphs with one edge type.
  - How to handle (directed) graphs with multiple edge types (a.k.a heterogeneous graphs)?

- Heterogeneous Graphs
  - Relational GCNs
  - Knowledge Graphs
  - Embeddings for KG Completion

Stanford CS224W: Heterogeneous Graphs and Relational GCN (RGCN)

CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

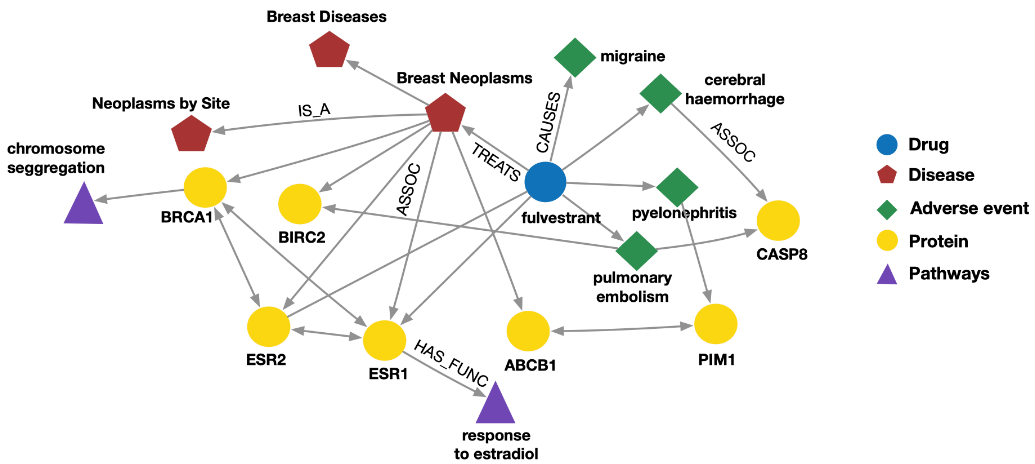
Heterogeneous Graphs

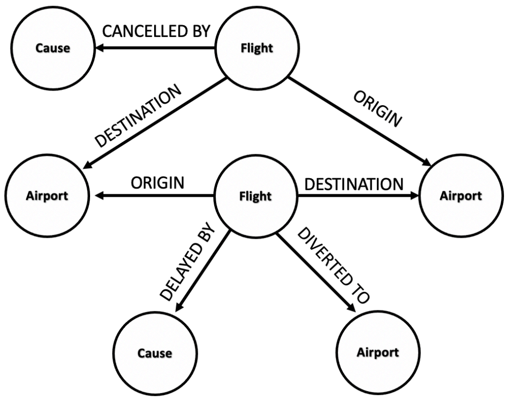
- **A heterogeneous graph is defined as**

$$\boldsymbol{G = (V, E, R, T)}$$

  - Nodes with node types $v_i \in V$
  - Edges with relation types $(v_i, r, v_j) \in E$
  - Node type $T(v_i)$
  - Relation type $r \in R$

Many Graphs are Heterogeneous Graphs (1)





Biomedical Knowledge Graphs

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

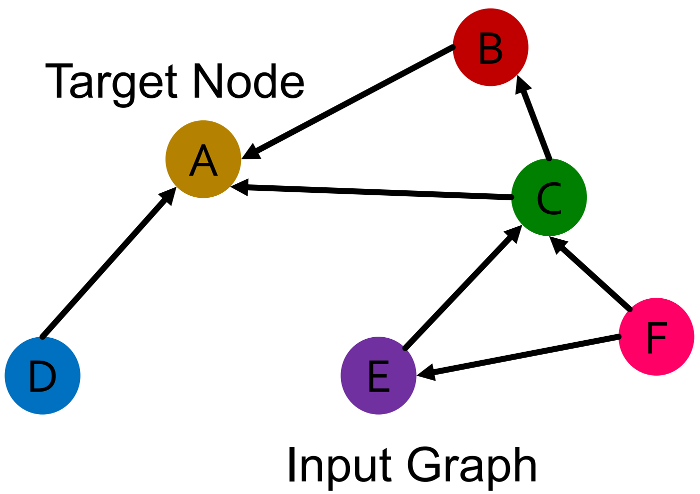
Event Graphs

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

Schlichtkrull et al., Modeling Relational Data with Graph Convolutional Networks, ESWC 2018

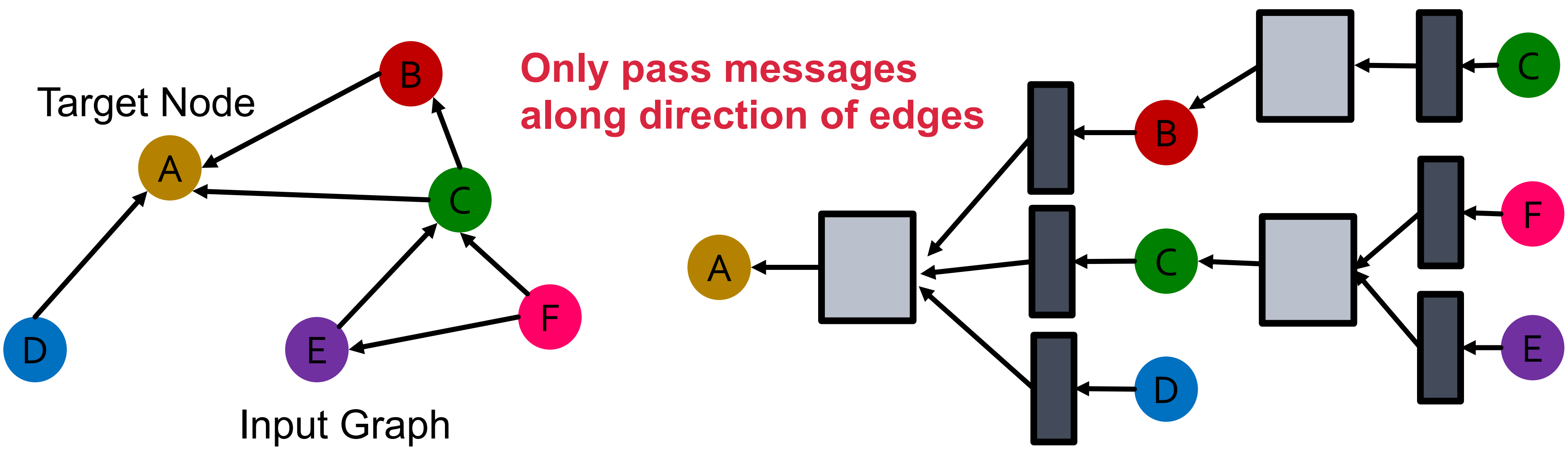
Relational GCN

- **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?



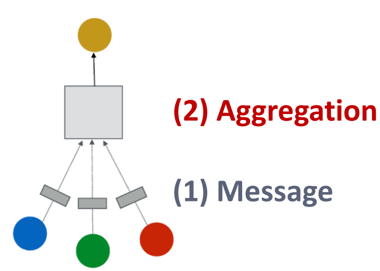
Relational GCN

- **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?



Recap: A Single GNN Layer

- **A single GNN layer:**
  - **(1) Message**: each node computes a message
    $$\mathbf{m}_u^{(l)} = \mathrm{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right), u \in \{N(v) \cup v\}$$
  - **(2) Aggregation**: aggregate messages from neighbors
    $$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l)}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}, \mathbf{m}_v^{(l)}\right)$$
  - **Nonlinearity (activation):** Adds expressiveness
    - Often written as $\sigma(\cdot)$: $\mathrm{ReLU}(\cdot)$, $\mathrm{Sigmoid}(\cdot)$, …
    - Can be added to **message or aggregation**



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

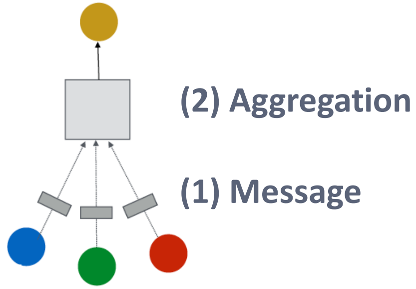
Recap: Classical GNN Layers: GCN (1)

- **(1) Graph Convolutional Networks (GCN)**

$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \sum_{u \in N(v)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$$

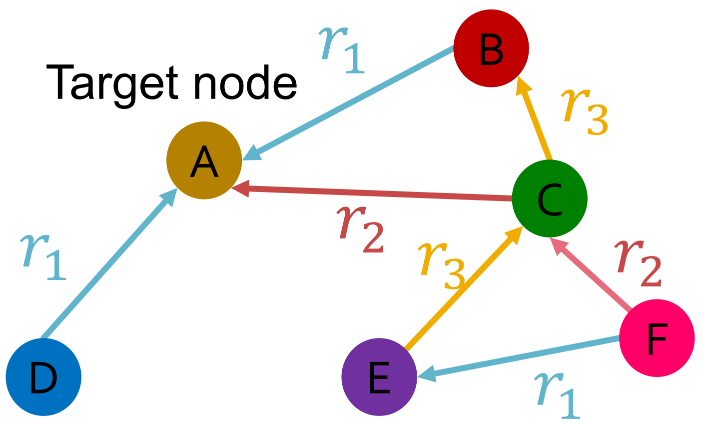
- **How to write this as Message + Aggregation?**

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



Relational GCN (1)

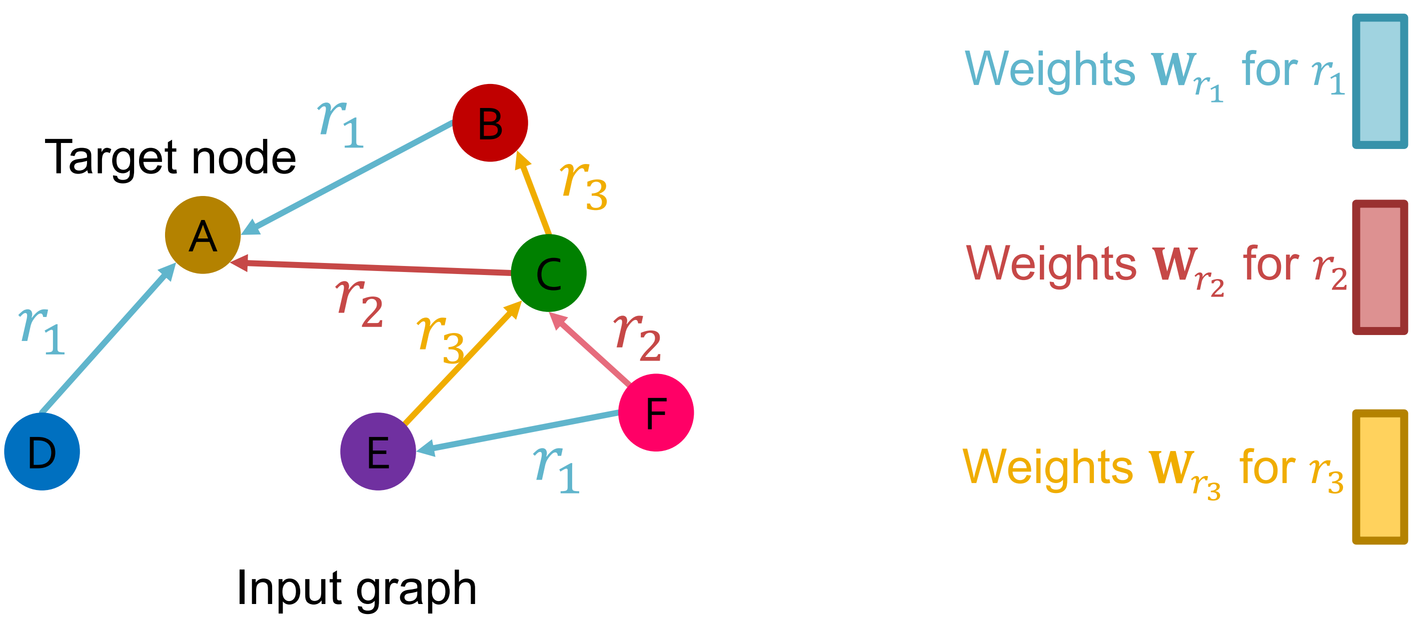
- **What if the graph has multiple relation types?**



Input graph

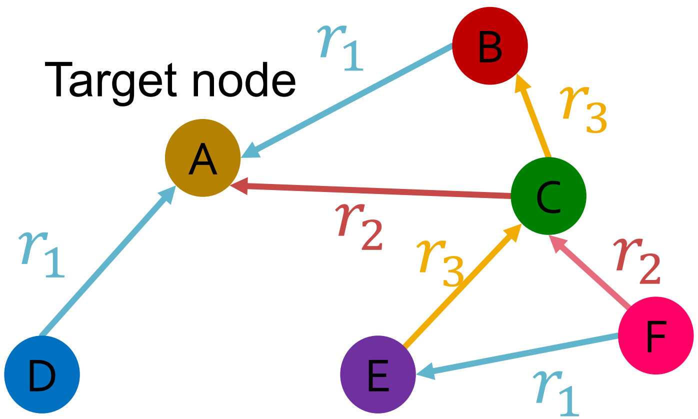
Relational GCN (2)

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

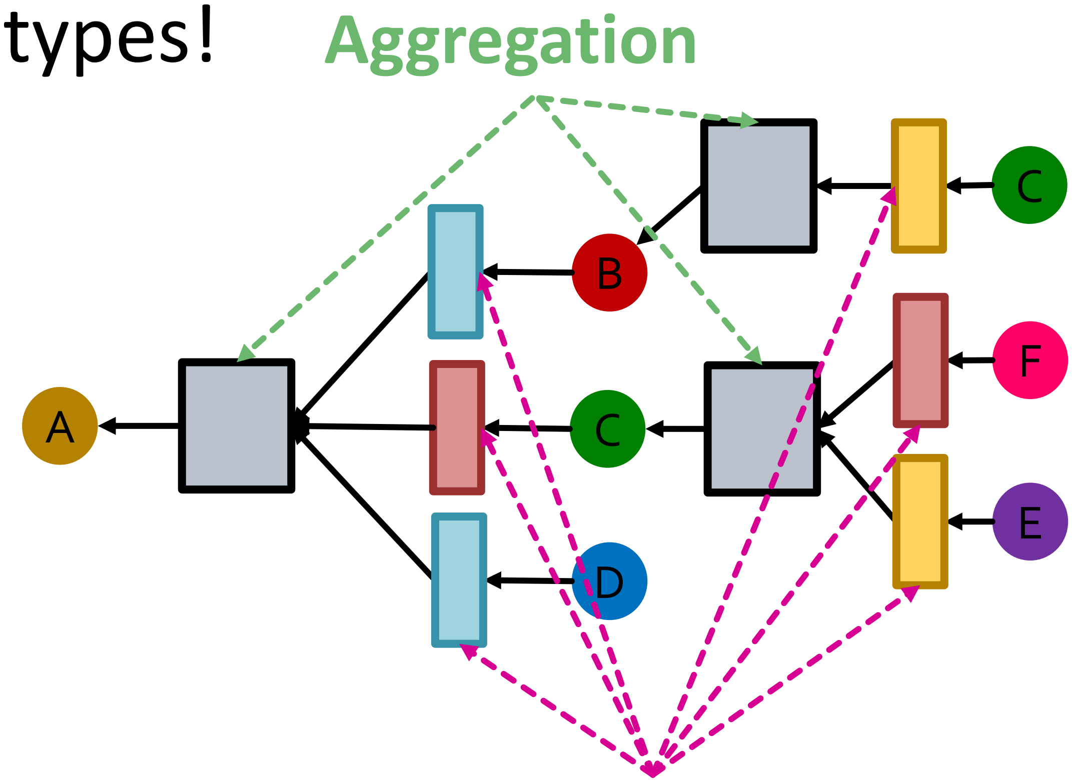


Relational GCN (3)

- What if this graph has **multiple relation types**?
- Use different neural network weights for different relation types!



Input graph



Relational GCN: Definition

- Relational GCN (RGCN):

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{r \in R} \sum_{u \in N_v^r} \frac{1}{c_{v,r}} \mathbf{W}_r^{(l)} \mathbf{h}_u^{(l)} + \mathbf{W}_0^{(l)} \mathbf{h}_v^{(l)}\right)$$

- How to write this as Message + Aggregation?
- Message:
  - Each neighbor of a given relation:

$$\mathbf{m}_{u,r}^{(l)} = \frac{1}{c_{v,r}} \mathbf{W}_r^{(l)} \mathbf{h}_u^{(l)}$$

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

  - Self-loop:

$$\mathbf{m}_v^{(l)} = \mathbf{W}_0^{(l)} \mathbf{h}_v^{(l)}$$

- Aggregation:
  - Sum over messages from neighbors and self-loop, then apply activation
  - $\mathbf{h}_v^{(l+1)} = \sigma\left(\mathrm{Sum}\left(\left\{\mathbf{m}_{u,r}^{(l)}, u \in \{N(v)\} \cup \{v\}\right\}\right)\right)$

RGCN: Scalability

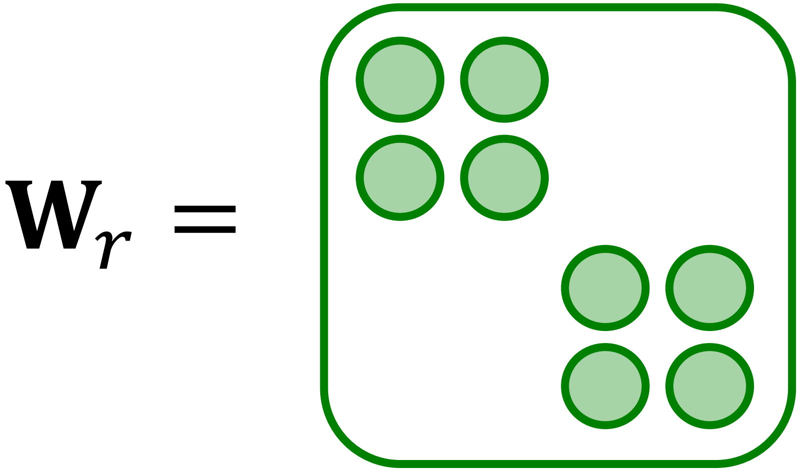
- Each relation has L matrices: $\mathbf{W}_r^{(1)}, \mathbf{W}_r^{(2)} \cdots \mathbf{W}_r^{(L)}$
- The size of each $\mathbf{W}_r^{(l)}$ is $d^{(l+1)} \times d^{(l)}$

$d^{(l)}$ is the hidden dimension in layer l

- **Rapid # parameters growth w.r.t # relations!**
  - **Overfitting becomes an issue**
- **Two methods to regularize the weights $\mathbf{W}_r^{(l)}$**
  - **(1)** Use block diagonal matrices
  - **(2)** Basis/Dictionary learning

(1) Block Diagonal Matrices

- **Key insight**: make the weights **sparse**!
- Use **block diagonal matrices** for $\mathbf{W}_r$



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 $B \times \frac{d^{(l+1)}}{B} \times \frac{d^{(l)}}{B}$.

(2) Basis Learning

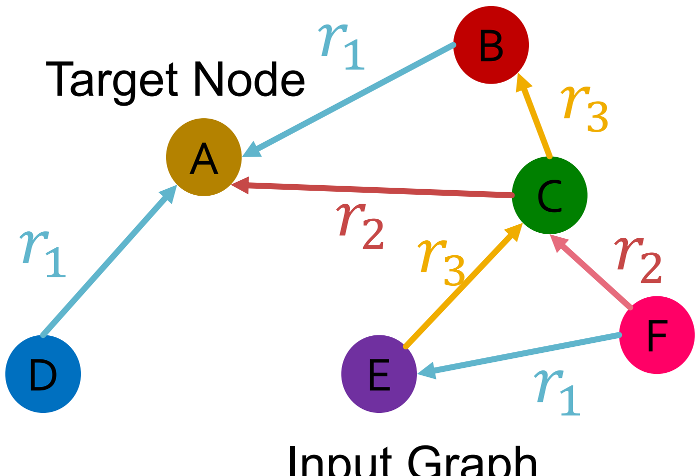
- **Key insight**: **Share weights** across different relations!

- Represent the matrix of each relation as a linear combination of basis transformations
  $\mathbf{W}_r = \sum_{b=1}^{B} a_{rb} \cdot \mathbf{V}_b$, where $\mathbf{V}_b$ is shared across all relations
  - $\mathbf{V}_b$ are the basis matrices
  - a_{rb} is the importance weight of matrix $\mathbf{V}_b$

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

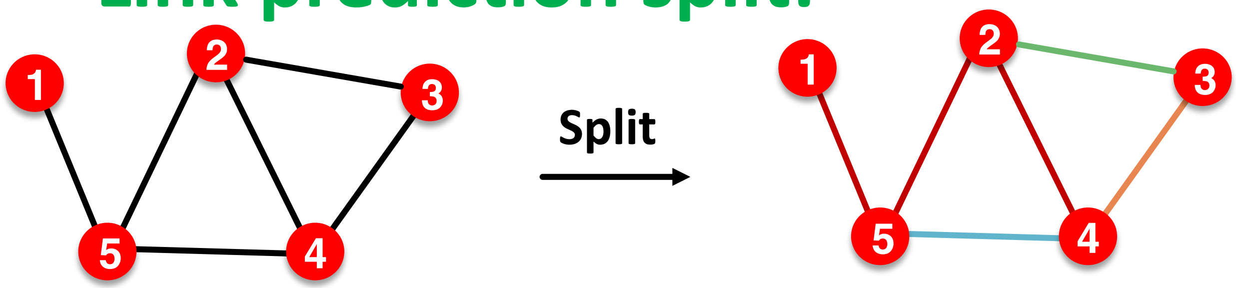
Example: Entity/Node Classification

- **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)**: $\mathbf{h}_A^{(L)} \in \mathbb{R}^k$, each item in $\mathbf{h}_A^{(L)}$ represents **the probability of that class**.



Example: Link Prediction

- **Link prediction split:**



The original graph

Split Graph with 4 categories of edges

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.

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

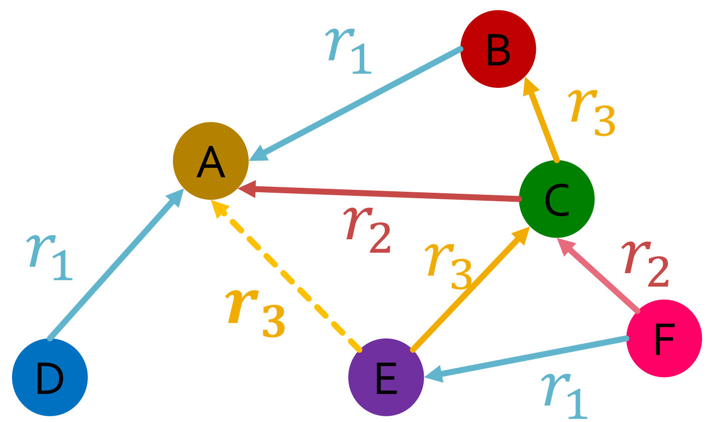
.....

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

RGCN for Link Prediction (1)

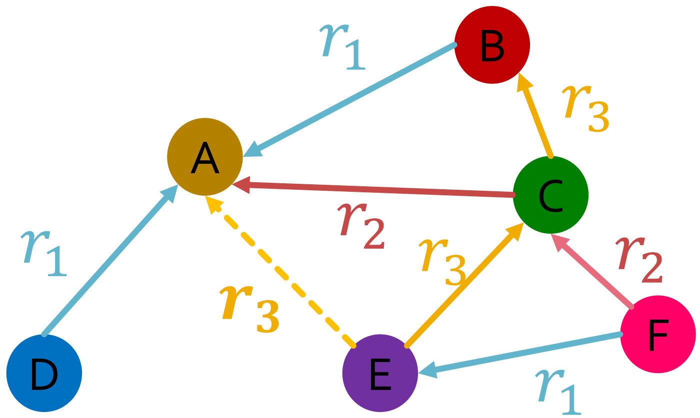
- **Assume (E, r_3, A) is Training supervision edge, all the other edges are Training message edges**
- **Use RGCN to score (E, r_3, A)!**
  - Take the final layer of E and A: $\mathbf{h}_E^{(L)}$ and $\mathbf{h}_A^{(L)} \in \mathbb{R}^d$
  - Relation-specific score function $f_r: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
    - One example $f_{r_1}(\mathbf{h}_E, \mathbf{h}_A) = \mathbf{h}_E^T \mathbf{W}_{r_1} \mathbf{h}_A$, $\mathbf{W}_{r_1} \in \mathbb{R}^{d \times d}$



Input Graph

RGCN for Link Prediction (2)

- **Training:**



Input Graph

1. **Use RGCN to score the training supervision edge (E, r_3, A)**
2. **Create negative edge by perturbing the supervision edge:**
   - **Corrupt the tail of (E, r_3, A)**

   e.g., (E, r_3, B), (E, r_3, D)

training supervision edges:(E, r_3, A)

training message edges: all the rest existing edges (solid lines)

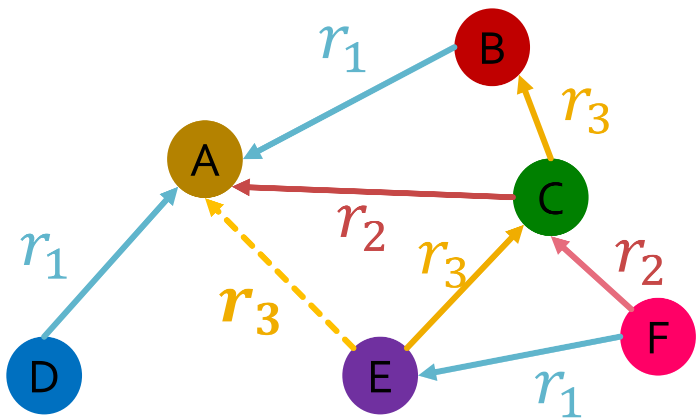
(1) Use **training message edges** to predict **training supervision edges**

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

RGCN for Link Prediction (3)

- **Training:**



Input Graph

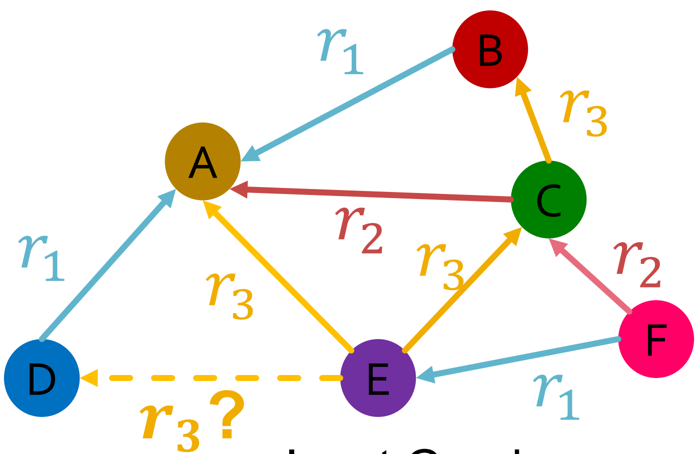
1. **Use RGCN to score the training supervision edge $(\boldsymbol{E}, \boldsymbol{r_3}, \boldsymbol{A})$**
2. **Create negative edge by perturbing the supervision edge $(\boldsymbol{E}, \boldsymbol{r_3}, \boldsymbol{B})$**
3. **Use GNN model to score negative edge**
4. **Optimize a standard cross entropy loss (as discussed in Lecture 6)**
   1. **Maximize the score of training supervision edge**
   2. **Minimize the score of negative edge**

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

Sigmoid function

RGCN for Link Prediction (4)

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



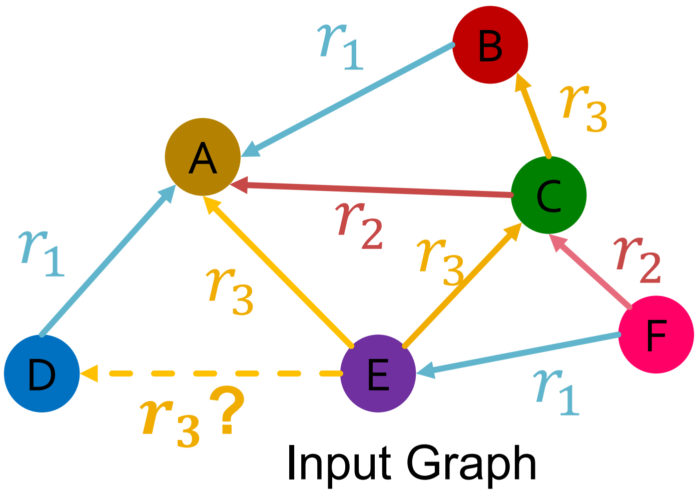
Input Graph

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 (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**

RGCN for Link Prediction (5)

- **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 (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: $\{(E, r_3, v) | v \in \{B, F\}\}$, since (E, r_3, A), (E, r_3, C) belong to training message edges & training supervision edges
3. Obtain the ranking 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

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.)

Stanford CS224W: Knowledge Graphs: KG Completion with Embeddings

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

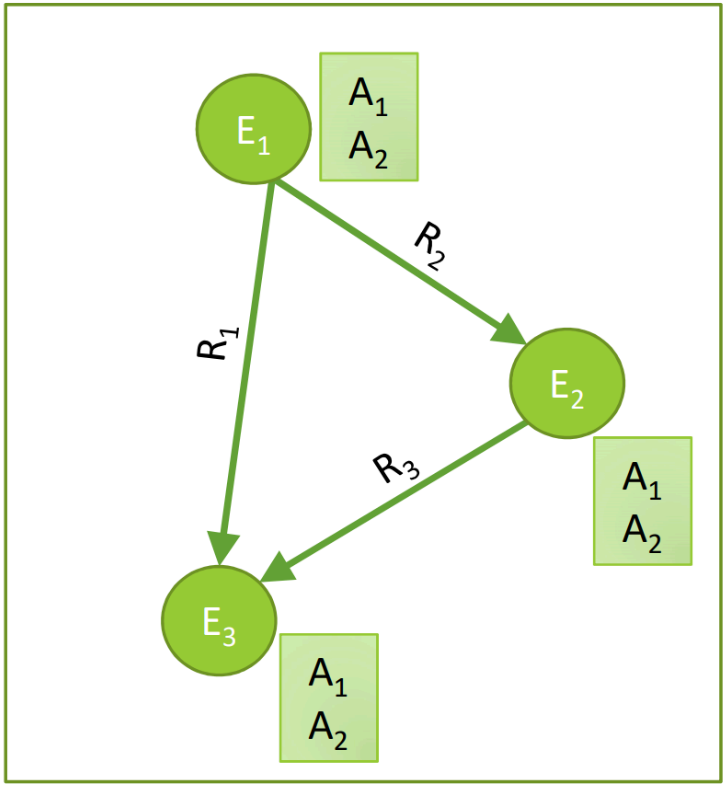
http://cs224w.stanford.edu

Knowledge Graphs (KG)

Knowledge in graph form:

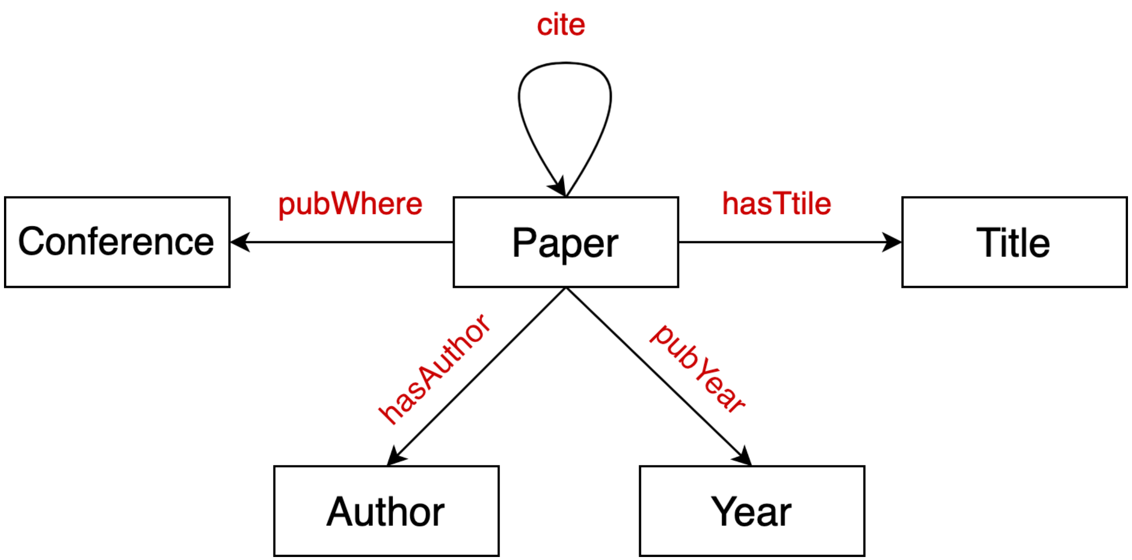
- Capture entities, types, and relationships

- 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**



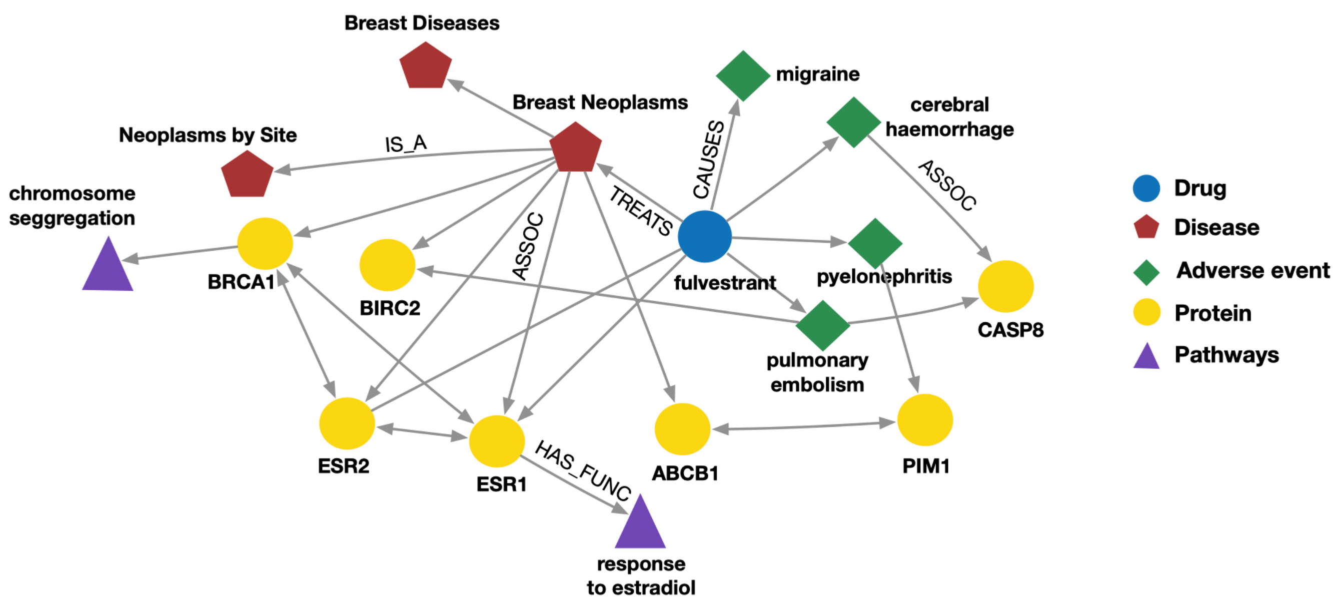
Example: Bibliographic Networks

- **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 Graphs 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**

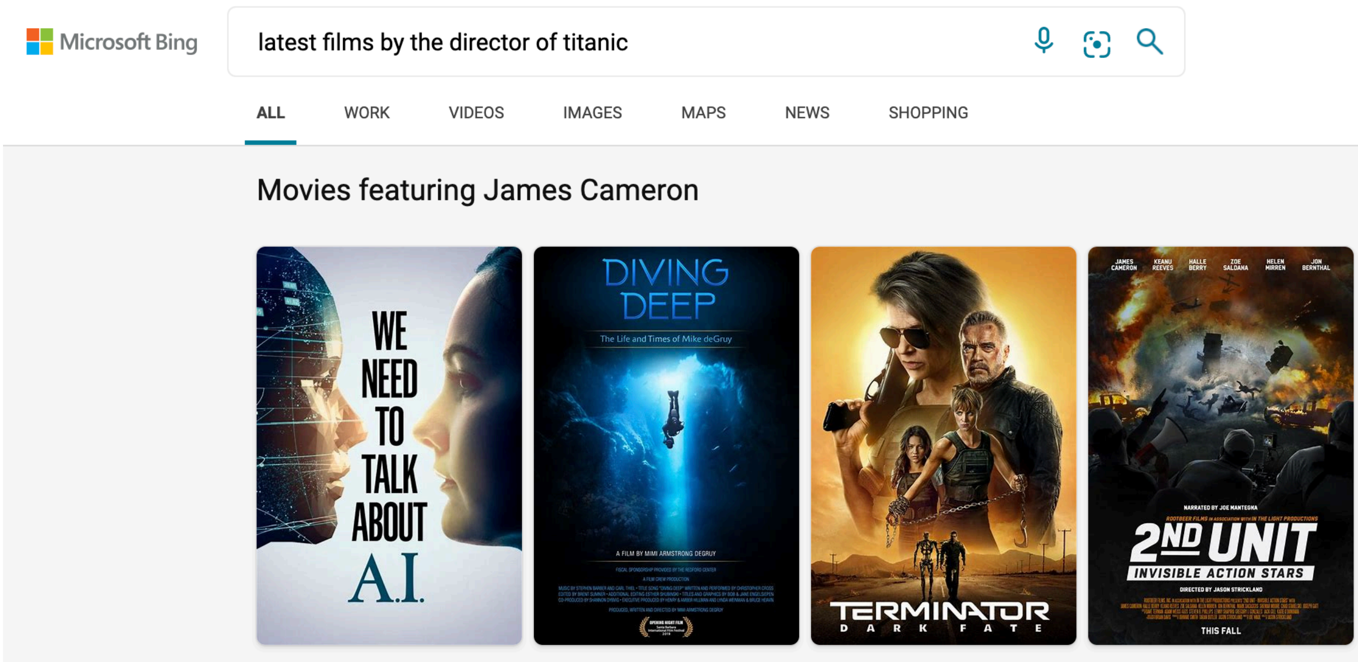


Image credit: Bing

Applications of Knowledge Graphs

- **Question answering and conversation agents**

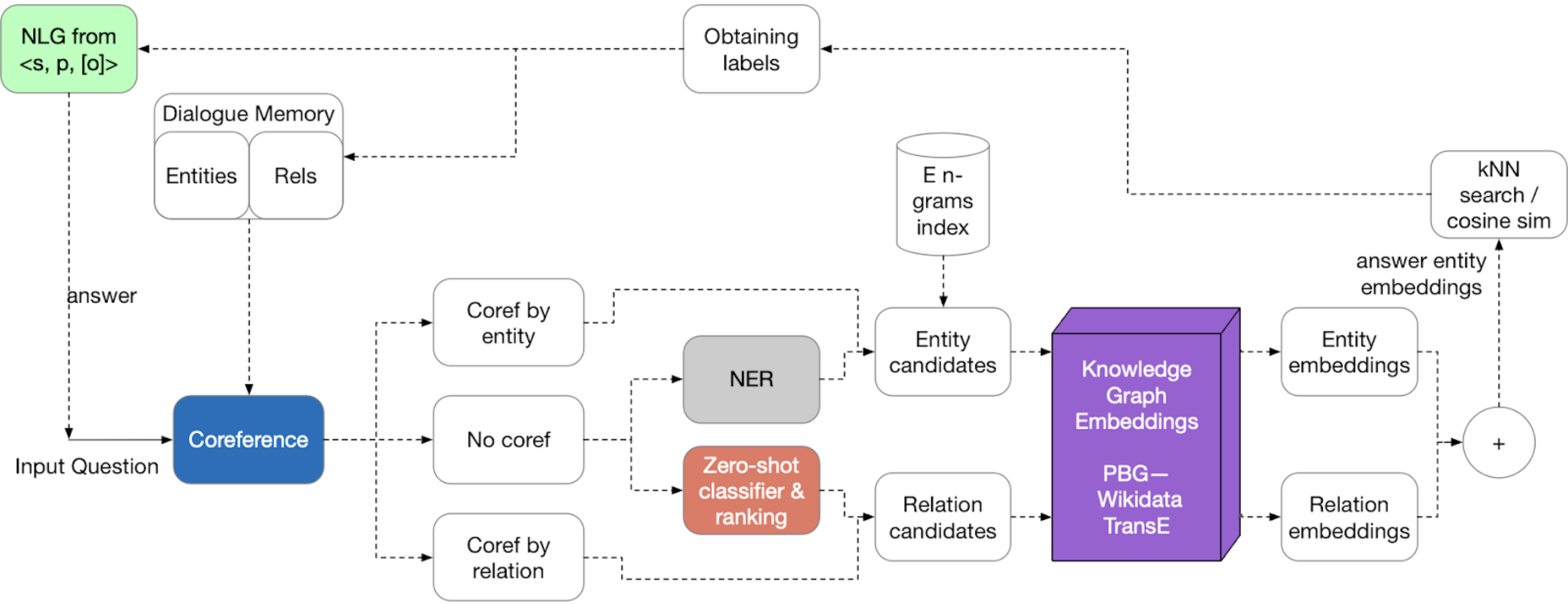


Image credit: [Medium]

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**
  - ~50 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!

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

Dataset	Entities	Relations	Total Edges
FB15k	14,951	1,345	592,213
FB15k-237	14,505	237	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.

Stanford CS224W: Knowledge Graph Completion: TransE, TransR, DistMul, ComplEx

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

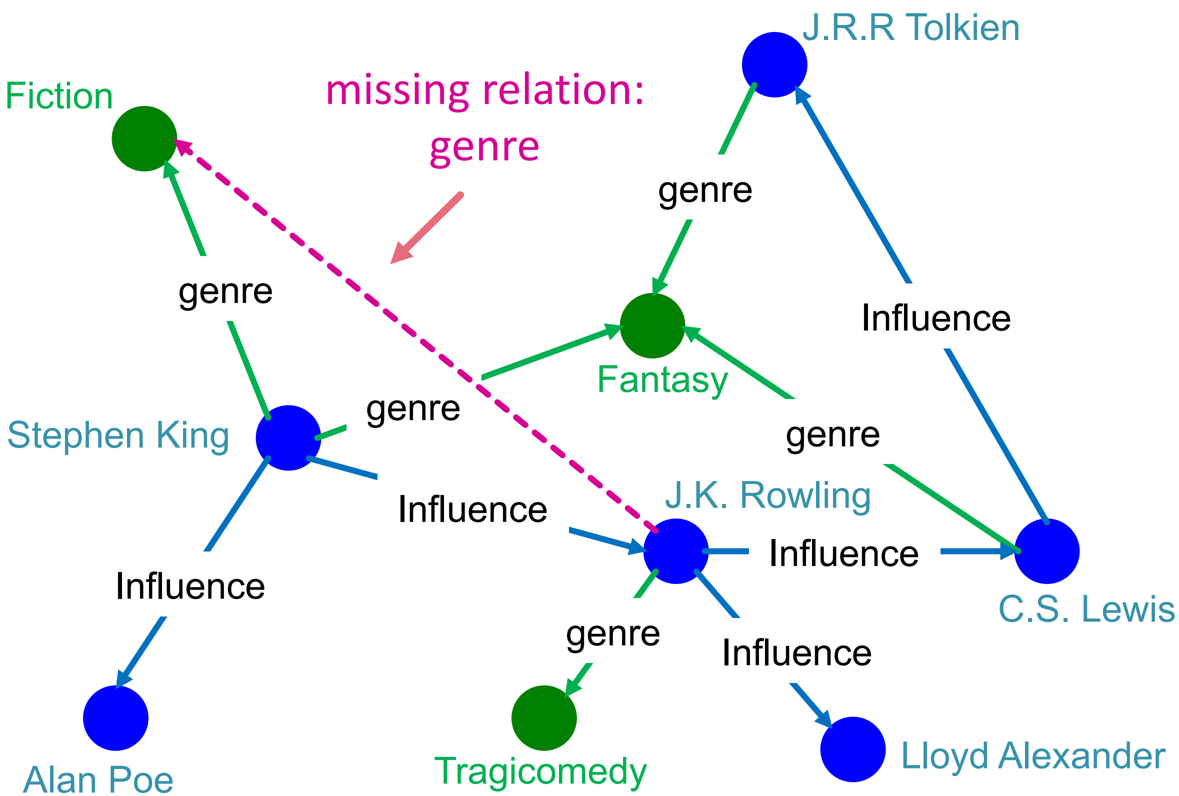
http://cs224w.stanford.edu

KG Completion Task

Given an enormous KG, can we complete the KG?

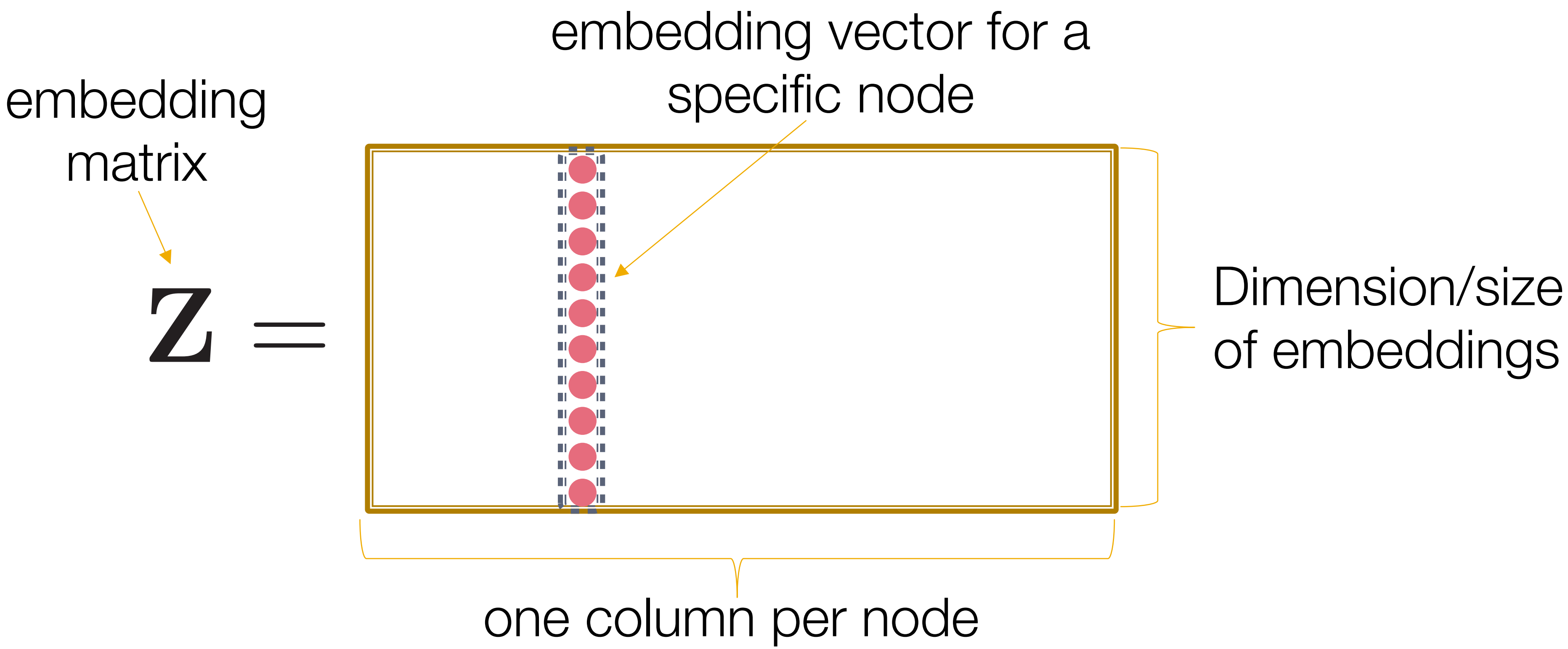
- For a given (**head**, **relation**), we predict missing **tails**.
  - (Note this is slightly different from link prediction task)

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



Recap: "Shallow" Encoding

- Simplest encoding approach: **encoder is just 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 the embedding/vector space $\mathbb{R}^d$.
    - Associate entities and relations with **shallow embeddings**
    - **Note we do not learn a GNN here!**
  - Given a true 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 closeness?

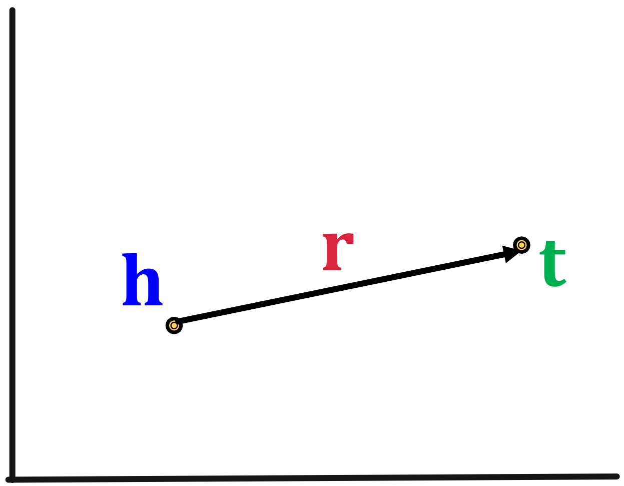
Bordes, et al., Translating embeddings for modeling multi-relational data, NeurIPS 2013.

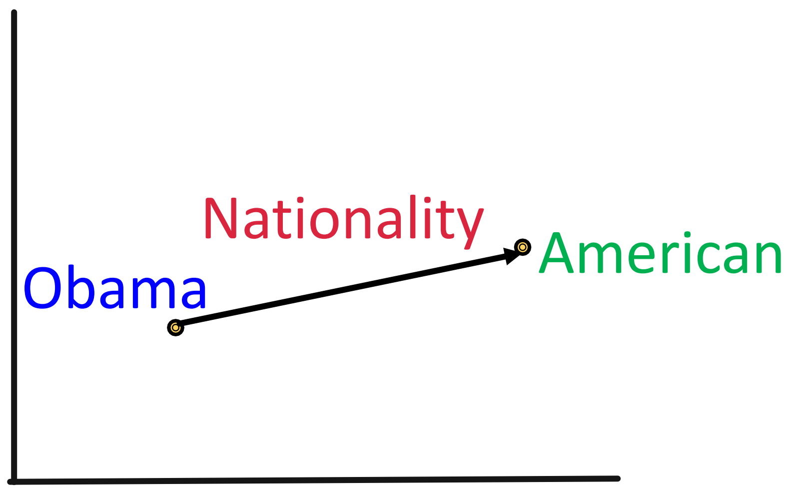
TransE

- **Translation Intuition**:
  For a triple (h, r, t), $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^d$,
  $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ if the given fact is true
  else $\mathbf{h} + \mathbf{r} \neq \mathbf{t}$

embedding vectors will appear in boldface

Scoring function: $f_r(h, t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||$





TransE Learning Algorithm

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 $e \in E$

Entities and relations are initialized uniformly, and normalized

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

Negative sampling with triplet that does not appear in the KG

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

end for

Update embeddings w.r.t. $$\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla \big[\gamma + \underbrace{d(\boldsymbol{h} + \boldsymbol{\ell}, \boldsymbol{t})}_{\text{positive sample}} - \underbrace{d(\boldsymbol{h'} + \boldsymbol{\ell}, \boldsymbol{t'})}_{\text{negative sample}}\big]_+$$

d represents distance (negative of score)

end loop

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 $(h, \text{"Roommate"}, t)$ exists in KG, then the edge $(t, \text{"Roommate"}, h)$ should also exist.
    - **Inverse relation:** If the edge $(h, \text{"Advisor"}, t)$ exists in KG, then the edge $(t, \text{"Advisee"}, h)$ should also exist.
- Can we **categorize** these relation patterns?
- **Are KG embedding methods (e.g., TransE) expressive enough to model these patterns?**

Relation Patterns

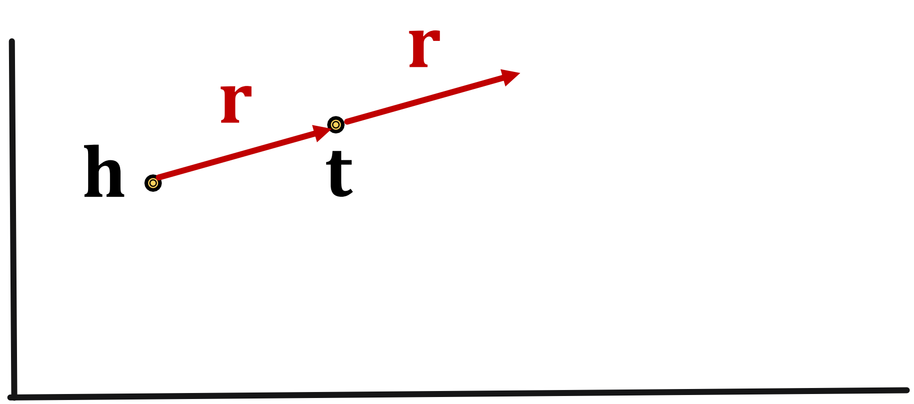
- **Symmetric (Antisymmetric) Relations:**
  $$r(h,t) \Rightarrow r(t,h) \;\; (r(h,t) \Rightarrow \neg r(t,h)) \quad \forall h,t$$
  - **Example**:
    - Symmetric: Family, Roommate
    - Antisymmetric: Hypernym
- **Inverse Relations:**
  $$r_2(h,t) \Rightarrow r_1(t,h)$$
  - **Example** : (Advisor, Advisee)
- **Composition (Transitive) Relations:**
  $$r_1(x,y) \wedge r_2(y,z) \Rightarrow r_3(x,z) \quad \forall x,y,z$$
  - **Example**: My mother's husband is my father.
- **1-to-N relations:**
  $r(h,t_1), r(h,t_2), \ldots, r(h,t_n)$ are all True.
  - **Example**: r is "StudentsOf"

Antisymmetric Relations in TransE

- **Antisymmetric Relations:**

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

  - **Example**: Hypernym
- **TransE** can model antisymmetric relations ✓
  - $\mathbf{h} + \mathbf{r} = \mathbf{t}$, but $\mathbf{t} + \mathbf{r} \neq \mathbf{h}$

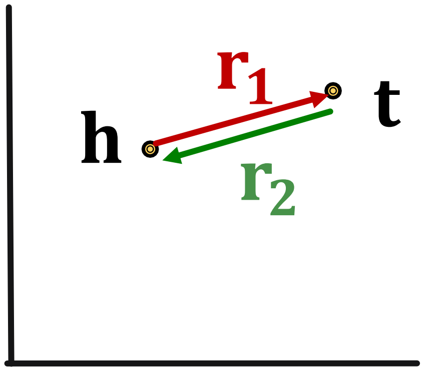


Inverse Relations in TransE

- **Inverse Relations:**

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

  - **Example** : (Advisor, Advisee)
- **TransE** can model inverse relations ✓
  - $\mathbf{h} + \mathbf{r}_2 = \mathbf{t}$, we can set $\mathbf{r}_1 = -\mathbf{r}_2$



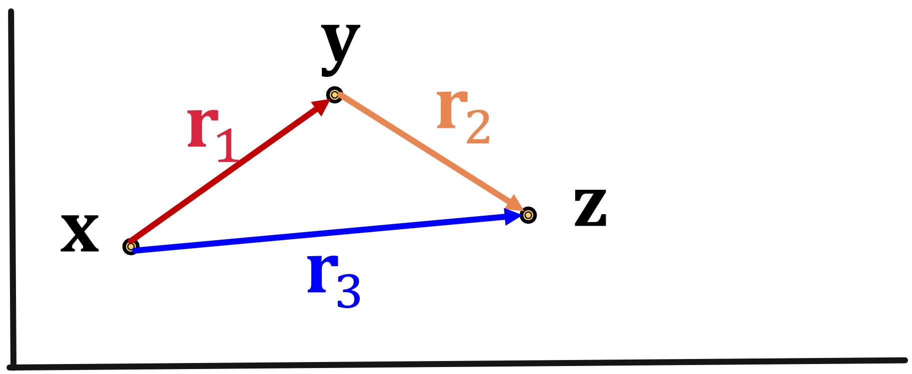
Composition in TransE

- **Composition (Transitive) Relations:**

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

  - **Example**: My mother's husband is my father.
- **TransE** can model composition relations ✓

$$\mathbf{r}_3 = \mathbf{r}_1 + \mathbf{r}_2$$



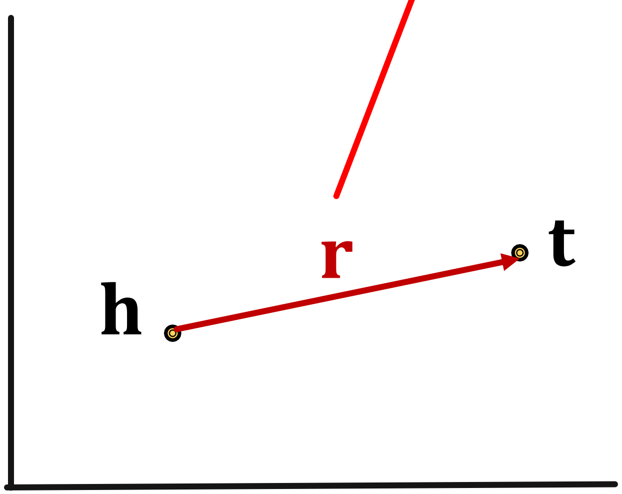
Limitation: Symmetric Relations

- **Symmetric Relations:**

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

  - **Example**: Family, Roommate
- **TransE cannot** model symmetric relations ✘

  only if $\mathbf{r} = 0,\ \mathbf{h} = \mathbf{t}$



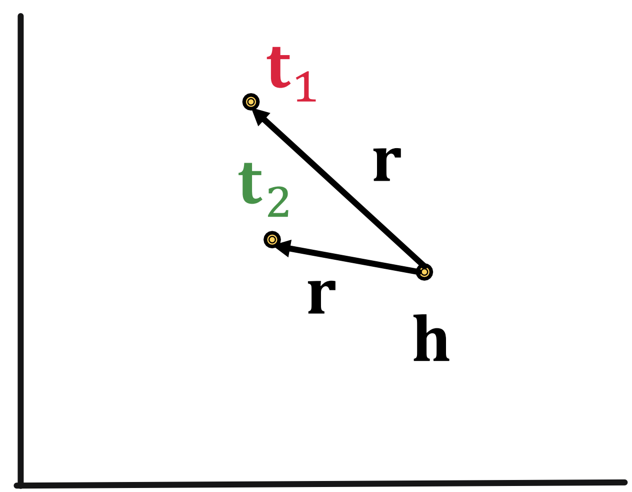
For all h, t that satisfy $r(h,t)$, $r(t,h)$ is also True, which means $||\mathbf{h} + \mathbf{r} - \mathbf{t}|| = 0$ and $||\mathbf{t} + \mathbf{r} - \mathbf{h}|| = 0$. Then $\mathbf{r} = 0$ and $\mathbf{h} = \mathbf{t}$, however h and t are two different entities and should be mapped to different locations.

Limitation: 1-to-N Relations

- **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 ✘
  - $\mathbf{t}_1$ and $\mathbf{t}_2$ will map to the same vector, although they are different entities

- $\mathbf{t}_1 = \mathbf{h} + \mathbf{r} = \mathbf{t}_2$
- $\mathbf{t}_1 \neq \mathbf{t}_2$

contradictory!



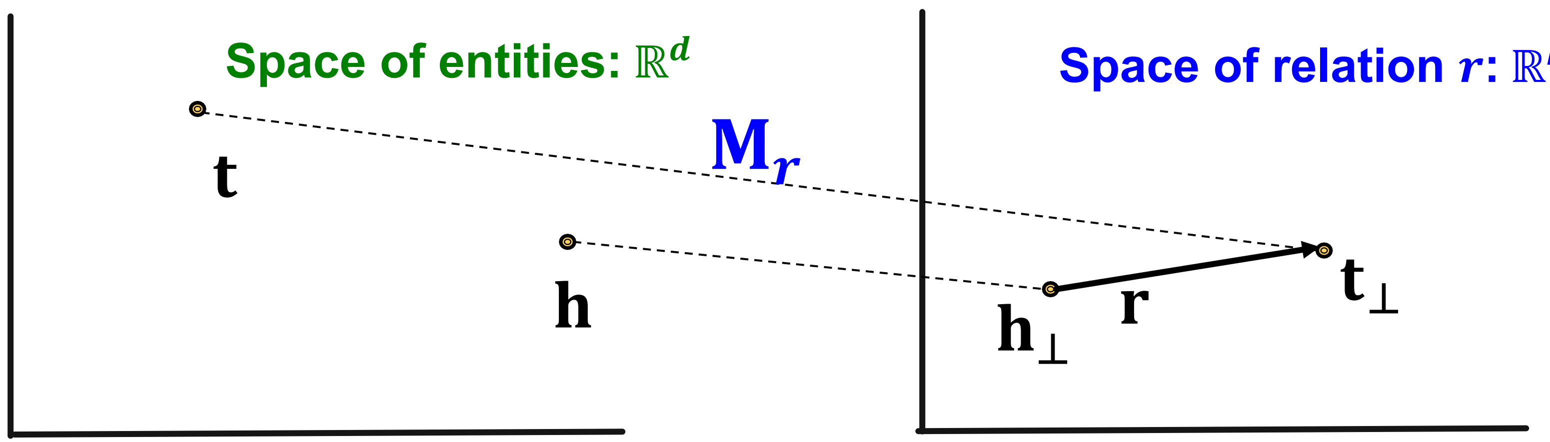
TransR

- **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 $\mathbf{M}_r \in \mathbb{R}^{k\times d}$ 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 $\mathbf{r} \in \mathbb{R}^k$ with $\mathbf{M}_r \in \mathbb{R}^{k \times d}$ as the **projection matrix**.
- $\mathbf{h}_\perp = \mathbf{M}_r \mathbf{h}, \ \mathbf{t}_\perp = \mathbf{M}_r \mathbf{t}$
- **Score function**: $f_r(h, t) = -||\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp||$

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



Symmetric Relations in TransR

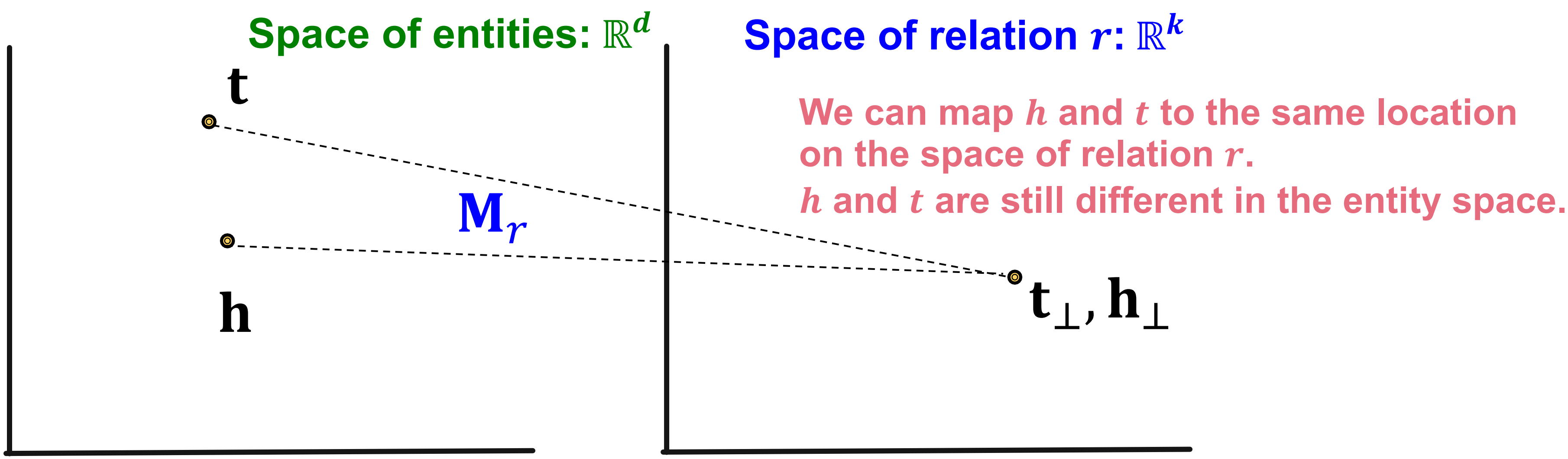
- **Symmetric Relations:**

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

  - **Example**: Family, Roommate
- **TransR** can model symmetric relations

$$\mathbf{r} = 0,\ \mathbf{h}_\perp = \mathbf{M}_r\mathbf{h} = \mathbf{M}_r\mathbf{t} = \mathbf{t}_\perp \checkmark$$

Note different symmetric relations may have different $\mathbf{M}_r$



Antisymmetric Relations in TransR

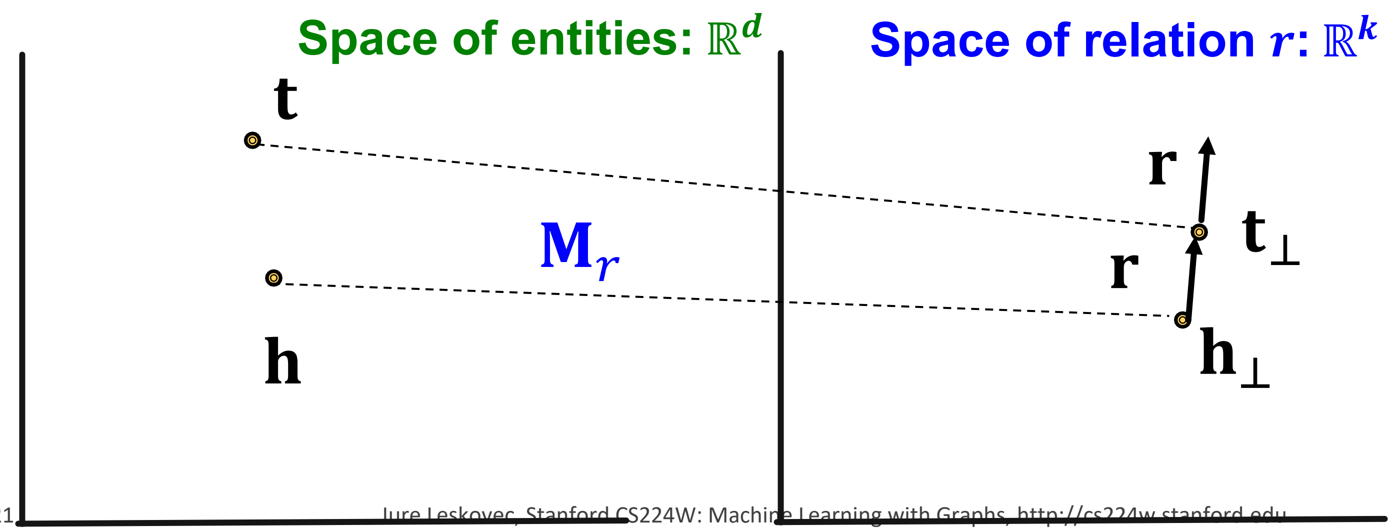
- **Antisymmetric Relations:**

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

  - **Example**: Hypernym
- **TransR** can model antisymmetric relations

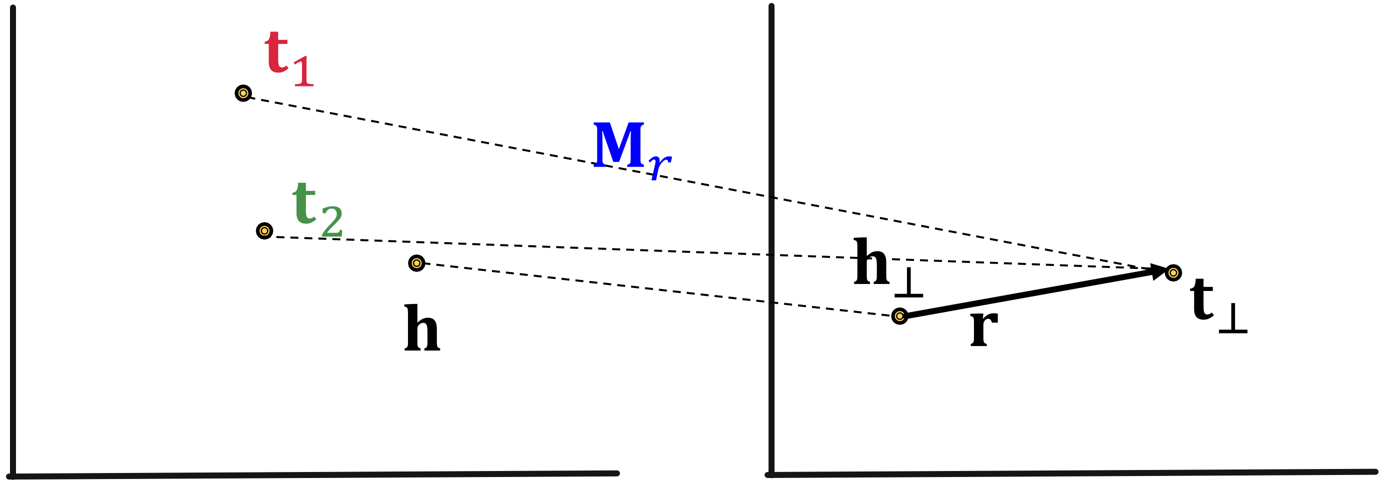
$$\mathbf{r} \neq 0, \mathbf{M}_r\mathbf{h} + \mathbf{r} = \mathbf{M}_r\mathbf{t},$$

$$\text{Then } \mathbf{M}_r\mathbf{t} + \mathbf{r} \neq \mathbf{M}_r\mathbf{t} \checkmark$$



1-to-N Relations in TransR

- **1-to-N Relations:**
  - **Example**: If (h, r, t_1) and (h, r, t_2) exist in the knowledge graph.
- **TransR** can model 1-to-N relations ✓
  - We can learn $\mathbf{M}_r$ so that $\mathbf{t}_\perp = \mathbf{M}_r \mathbf{t}_1 = \mathbf{M}_r \mathbf{t}_2$
  - Note that $\mathbf{t}_1$ does not need to be equal to $\mathbf{t}_2$!



Inverse Relations in TransR

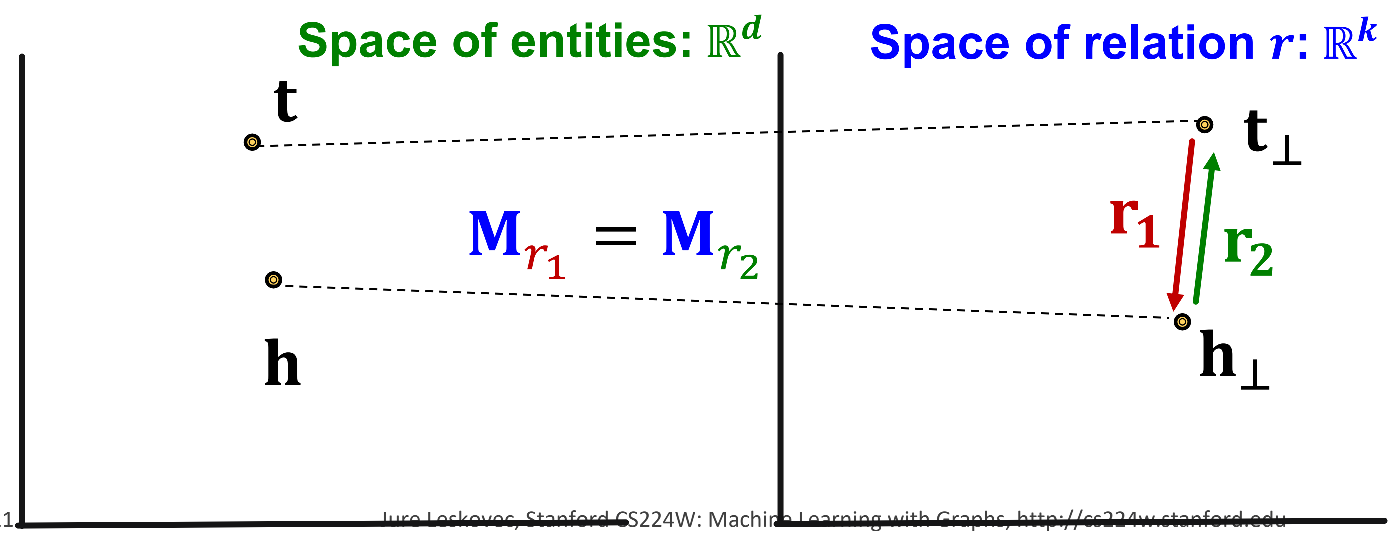
- **Inverse Relations:**

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

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

$$\mathbf{r}_2 = -\mathbf{r}_1, \mathbf{M}_{r_1} = \mathbf{M}_{r_2}$$

Then $\mathbf{M}_{r_1}\mathbf{t} + \mathbf{r}_1 = \mathbf{M}_{r_1}\mathbf{h}$ and $\mathbf{M}_{r_2}\mathbf{h} + \mathbf{r}_2 = \mathbf{M}_{r_2}\mathbf{t}$ ✓



Limitation: Composition Relations

- **Composition Relations:**

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

  - **Example**: My mother's husband is my father.
- **TransR cannot** model composition relations

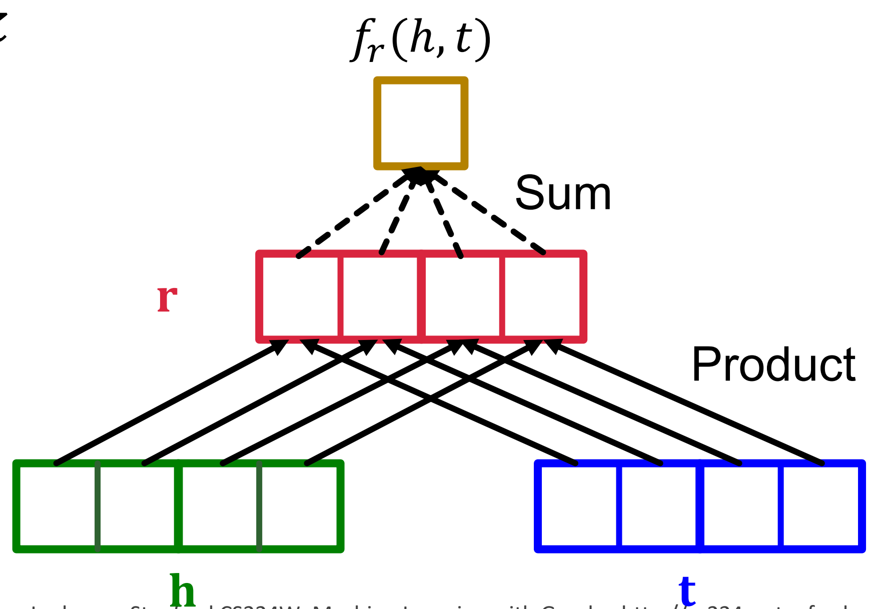
Each relation has a different space.
It is **not naturally compositional** for multiple relations! ×

Zero Distance

Intuition: The manifold $\{z \mid \exists y, f_{r_1}(x, y) = 0, f_{r_2}(y, z) = 0\}$ is **high dimensional** and may not be modeled using a single $\mathbf{M}_{r_3}$ and $\mathbf{r}_3$

New Idea: Bilinear Modeling

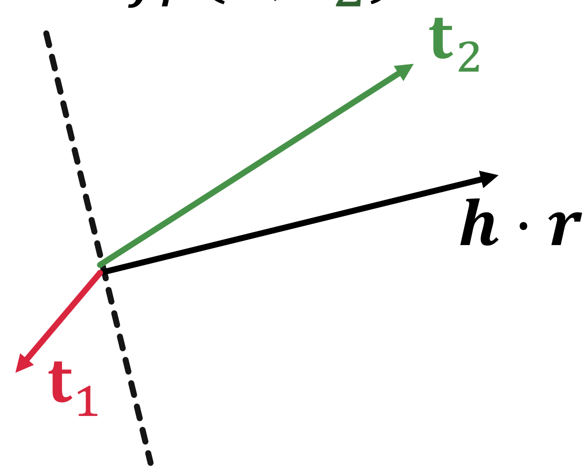
- **So far:** The scoring function $f_r(h, t)$ is **negative of L1 / L2 distance** in **TransE** and **TransR**
- Another line of KG embeddings adopt **bilinear modeling**
- **DistMult**: Entities and relations using vectors in $\mathbb{R}^k$
- **Score function:** $f_r(h, t) =< \mathbf{h}, \mathbf{r}, \mathbf{t} > = \sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i$
- $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$



DistMult

- **DistMult**: Entities and relations using vectors in $\mathbb{R}^k$
- **Score function:** $f_r(h,t) = <\mathbf{h},\mathbf{r},\mathbf{t}> = \sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i$
- $\mathbf{h},\mathbf{r},\mathbf{t} \in \mathbb{R}^k$
- **Intuition of the score function:** Can be viewed as a cosine similarity between $\mathbf{h} \cdot \mathbf{r}$ and $\mathbf{t}$
- **Example:**

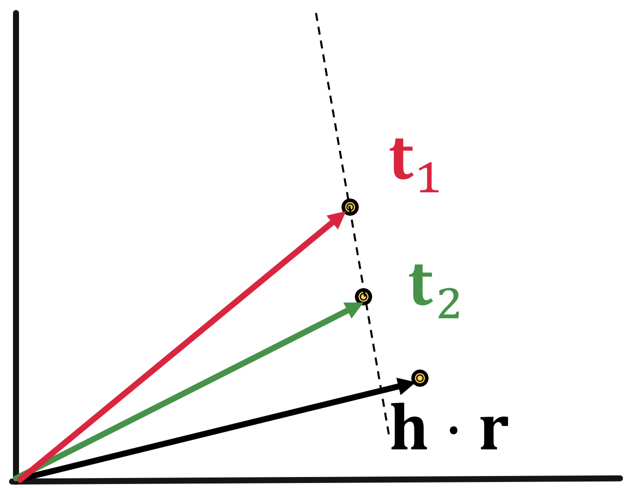
$$f_r(h,t_1) < 0, \qquad f_r(h,t_2) > 0$$



1-to-N Relations in DistMult

- **1-to-N Relations:**
  - **Example**: If (h, r, t_1) and (h, r, t_2) exist in the knowledge graph
- **Distmult** can model 1-to-N relations ✓

$$< \mathbf{h}, \mathbf{r}, \mathbf{t}_1 > = < \mathbf{h}, \mathbf{r}, \mathbf{t}_2 >$$



Symmetric Relations in DistMult

- **Symmetric Relations:**

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

  - **Example**: Family, Roommate
- **DistMult** can naturally model symmetric relations ✓

$$f_r(h,t) = <\mathbf{h},\mathbf{r},\mathbf{t}> = \sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i =$$

$$<\mathbf{t},\mathbf{r},\mathbf{h}> = f_r(t,h)$$

Limitation: Antisymmetric Relations

- **Antisymmetric Relations:**

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

  - **Example**: Hypernym
- **DistMult cannot** model antisymmetric relations

$$f_r(h,t) = <\mathbf{h},\mathbf{r},\mathbf{t}> = <\mathbf{t},\mathbf{r},\mathbf{h}> = f_r(t,h) \text{ ✘}$$

  - $r(h,t)$ and $r(t,h)$ always have same score!

Limitation: Inverse Relations

- **Inverse Relations:**

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

  - **Example** : (Advisor, Advisee)
- **DistMult cannot** model inverse relations ✘
  - If it does model inverse relations:

$$f_{r_2}(h,t) = <\mathbf{h}, \mathbf{r}_2, \mathbf{t}> = <\mathbf{t}, \mathbf{r}_1, \mathbf{h}> = f_{r_1}(t,h)$$

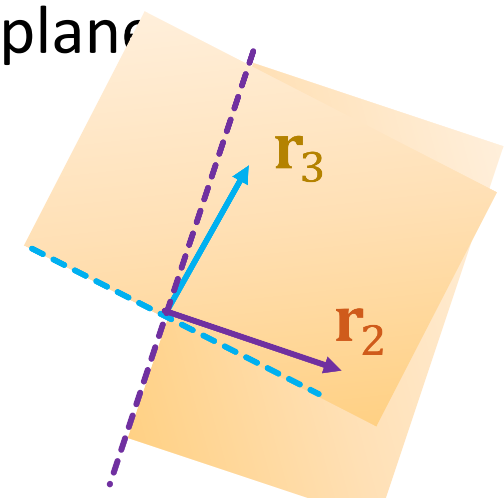
  - This means $\mathbf{r}_2 = \mathbf{r}_1$
  - But semantically this does not make sense: **The embedding of "Advisor" should not be the same with "Advisee".**

Limitation: Composition Relations

- **Composition Relations:**

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

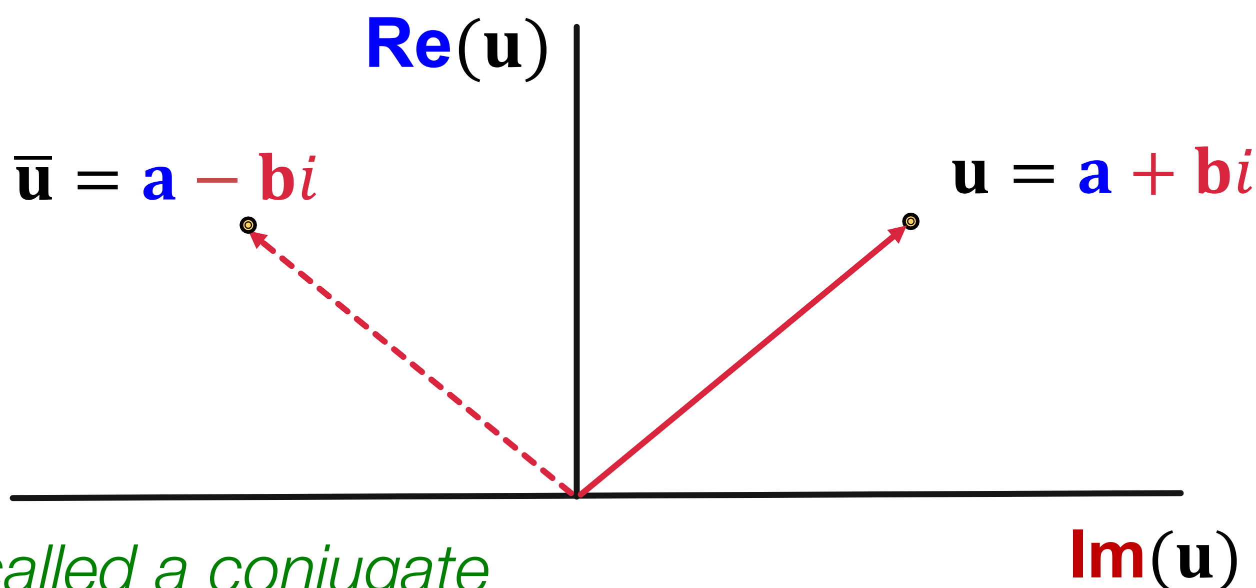
  - **Example**: My mother's husband is my father.
- **DistMult cannot** model composition relations ×
  - **Intuition: DistMult** defines a hyperplane for each (head, relation), the union of the hyperplane induced by multi-hops of relations, e.g., (r_1, r_2), cannot be expressed using a single hyperplane.



ComplEx

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

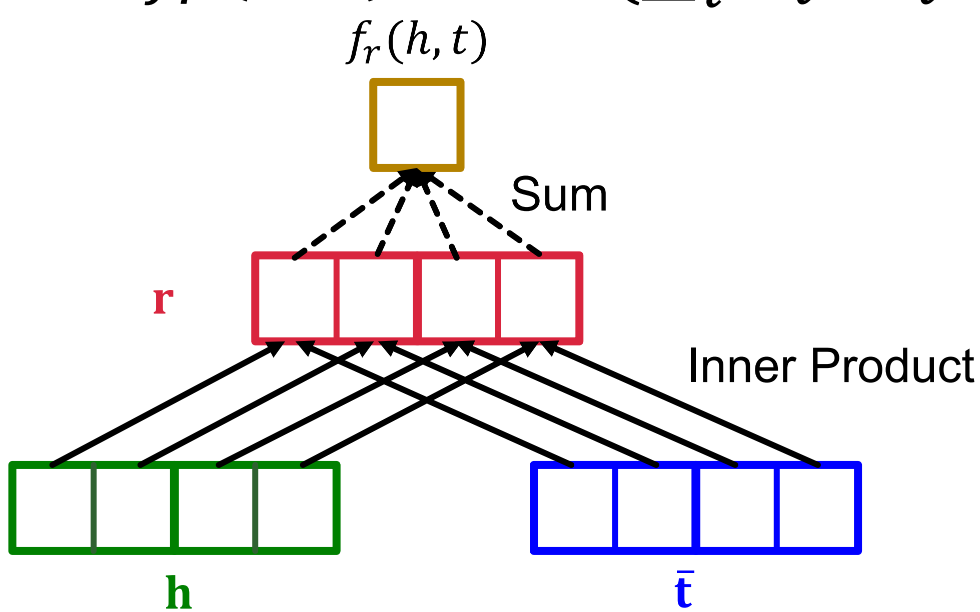
$\mathbf{u} \in \mathbb{C}^k$
$\mathbf{a} \in \mathbb{R}^k$
$\mathbf{b} \in \mathbb{R}^k$



$\bar{\mathbf{u}}$ is called a conjugate

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



Antisymmetric Relations in ComplEx

- **Antisymmetric Relations:**

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

  - **Example**: Hypernym
- **ComplEx** can model antisymmetric relations ✓
  - The model is expressive enough to learn
    - **High** $f_r(h,t) = \mathrm{Re}(\sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{t}}_i)$
    - **Low** $f_r(t,r) = \mathrm{Re}(\sum_i \boldsymbol{t}_i \cdot \mathbf{r}_i \cdot \bar{\boldsymbol{h}}_i)$

    Due to the asymmetric modeling using complex conjugate.

Symmetric Relations in ComplEx

- **Symmetric Relations:**

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

  - **Example**: Family, Roommate
- **ComplEx** can model symmetric relations ✓
  - When $\mathrm{Im}(\mathbf{r}) = 0$, 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 ✓
  - $\mathbf{r}_1 = \bar{\mathbf{r}}_2$
  - Complex conjugate of $\mathbf{r}_2 = \underset{\mathbf{r}}{\operatorname{argmax}} \operatorname{Re}(<\mathbf{h}, \mathbf{r}, \bar{\mathbf{t}}>)$ is exactly $\mathbf{r}_1 = \underset{\mathbf{r}}{\operatorname{argmax}} \operatorname{Re}(<\mathbf{t}, \mathbf{r}, \bar{\mathbf{h}}>)$.

Composition and 1-to-N

- **Composition Relations:**

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

  - **Example**: My mother's husband is my father.
- **1-to-N Relations:**
  - **Example**: If (h, r, t_1) and (h, r, t_2) exist in the knowledge graph
- **ComplEx** share the same property with **DistMult**
  - **Cannot model composition relations**
  - **Can model 1-to-N relations**

Expressiveness of All Models

- Properties and expressive power of different KG completion methods:

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

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 of the second part

- 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

Asymmetric Relations in ComplEx

- END