COMP4222 Machine Learning with Structured Data

Heterogeneous Graphs and Knowledge Graphs

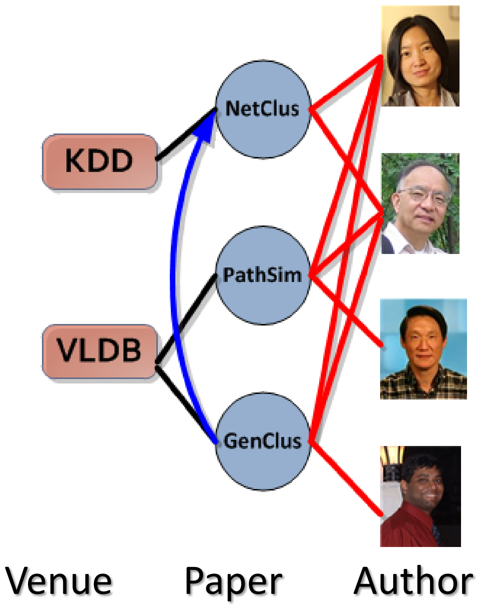
Instructor: Yangqiu Song

Slides credits: Jure Laskovec, Yizhou Sun, Evgeniy Gabrilovich

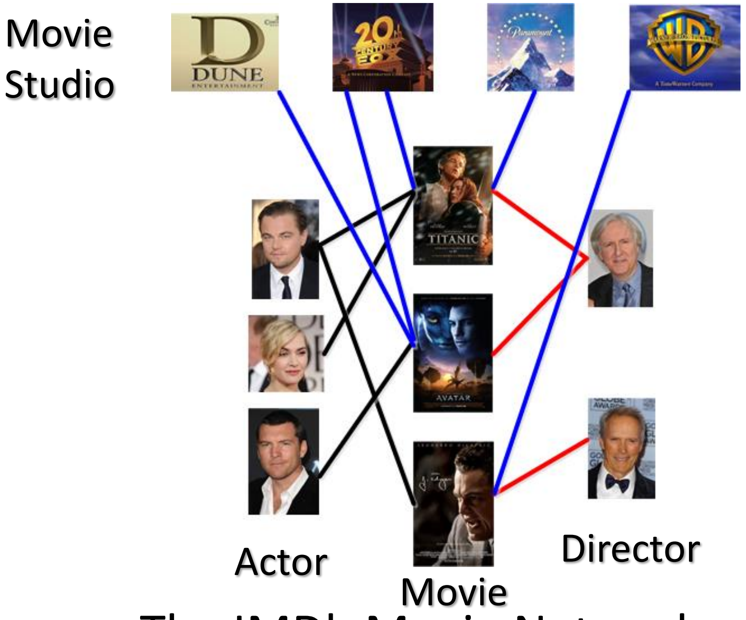
What Are Information Networks?

- A network where each node represents an entity (e.g., user in a social network) and each link (e.g., friendship) a relationship between entities.
  - Nodes/links may have attributes, labels, and weights.
  - Links may carry rich semantic information.

Heterogeneous Information Networks



DBLP Bibliographic Network

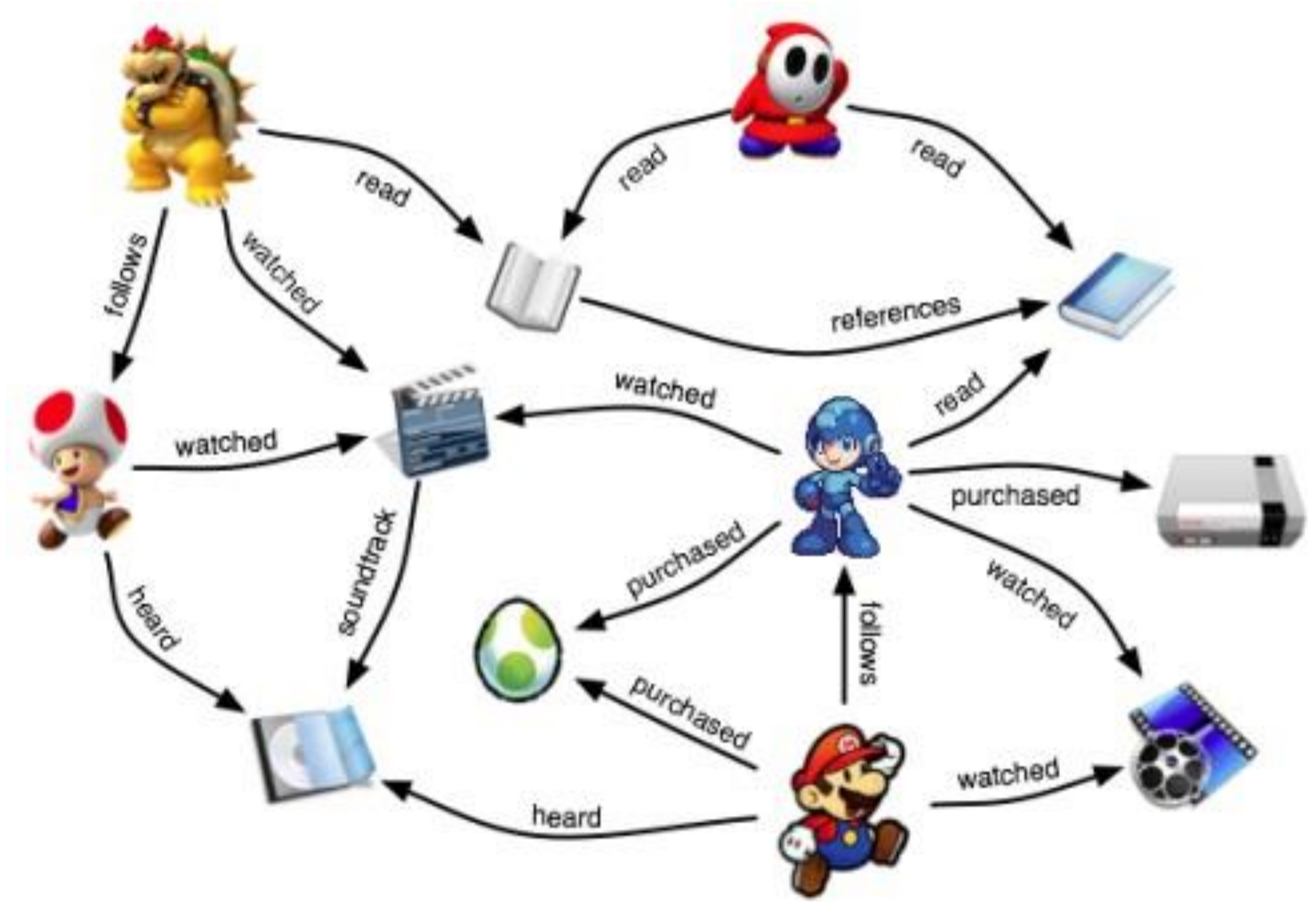


The IMDb Movie Network

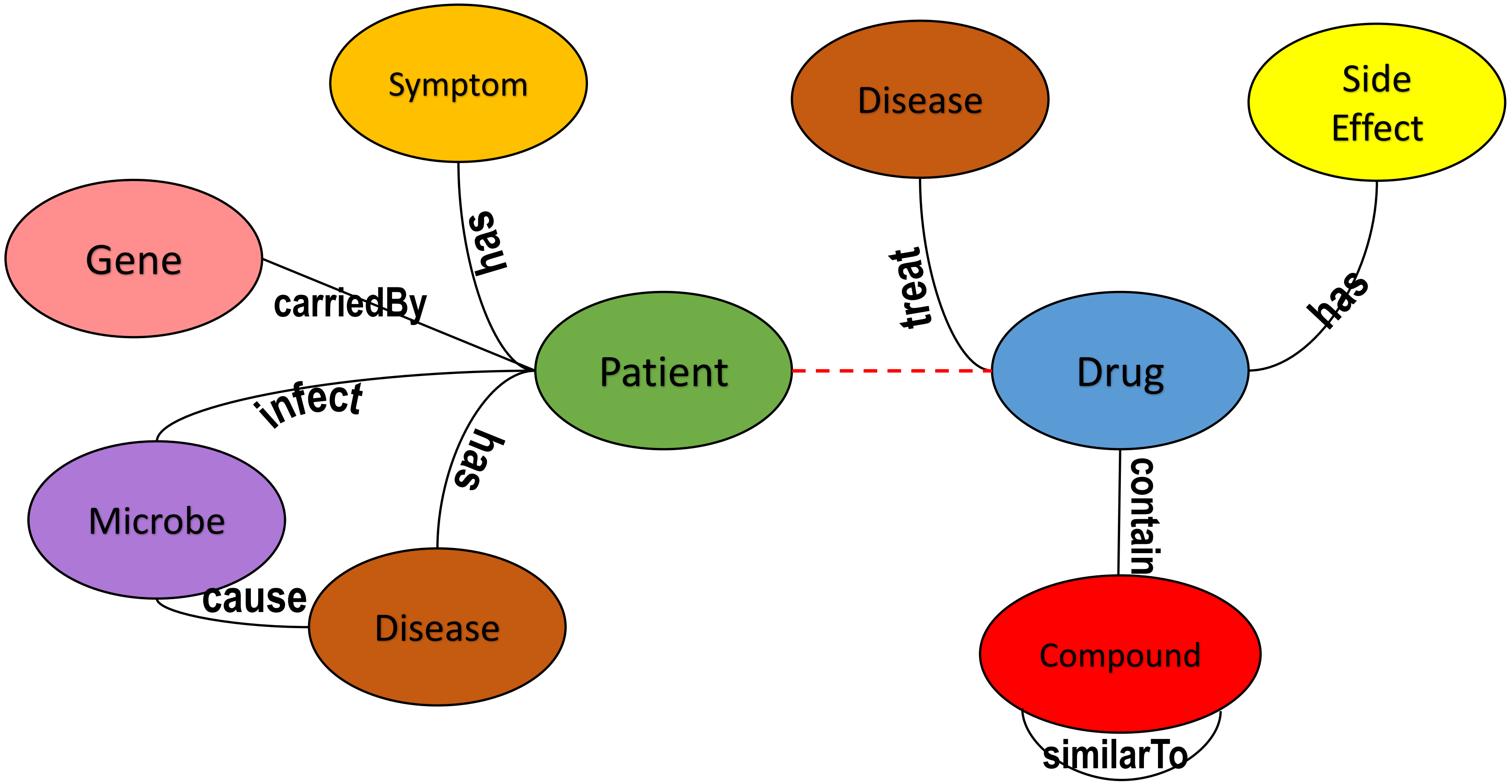
The Facebook Network

1. Multiple entity types and link types
2. *New problems* are emerging in heterogeneous networks!

We are living in a connected world!



Even in Biomedical Domain

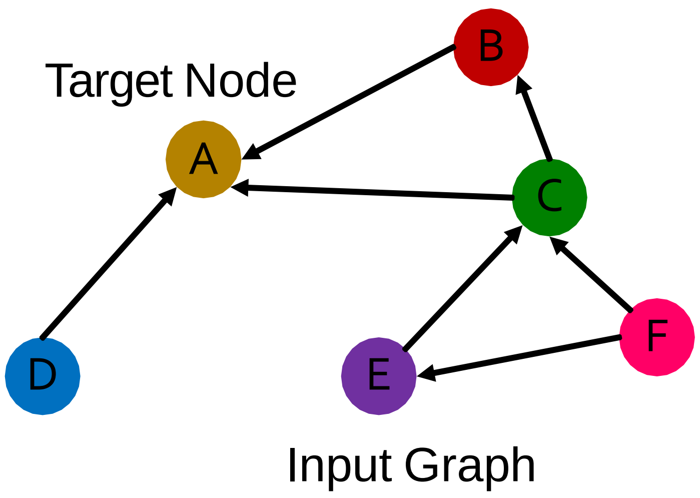


Heterogeneous Graphs

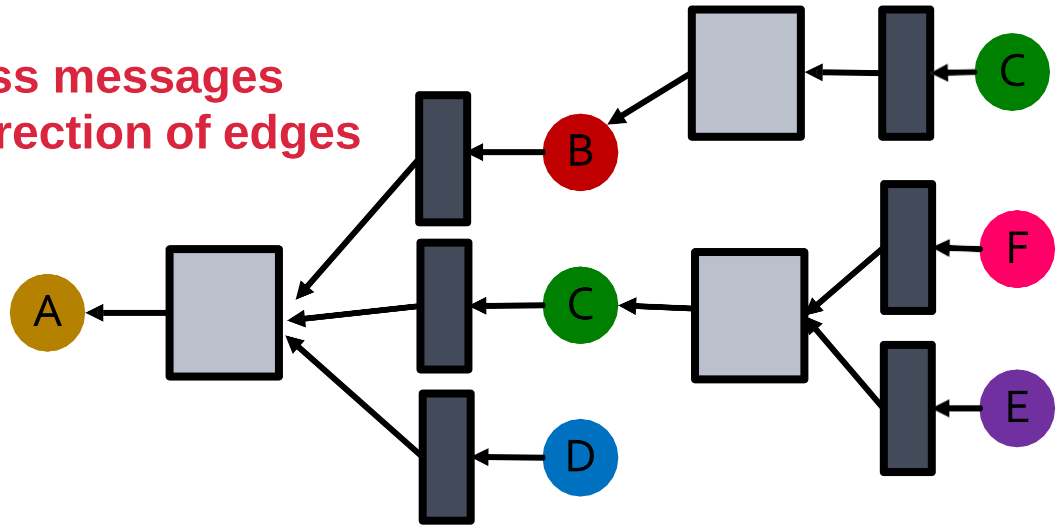
- **A heterogeneous graph is defined as**
  - $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{R}, \boldsymbol{T})$
    - Nodes with node types $v_i \in \boldsymbol{V}$
    - Edges with relation types $(v_i, r, v_j) \in E$
    - Node type $T(v_i)$
    - Relation type $r \in R$

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?

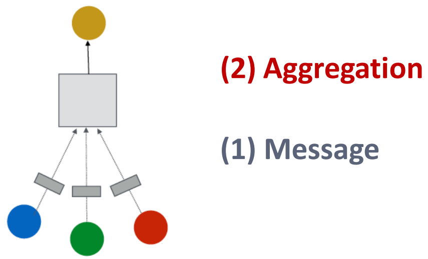


Only pass messages along direction of edges



Recap: Message Passing

- **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**



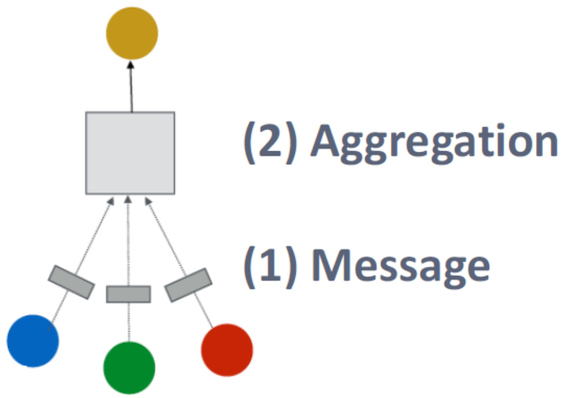
Recap: GCN

- **Graph Convolutional Networks (GCN)**

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

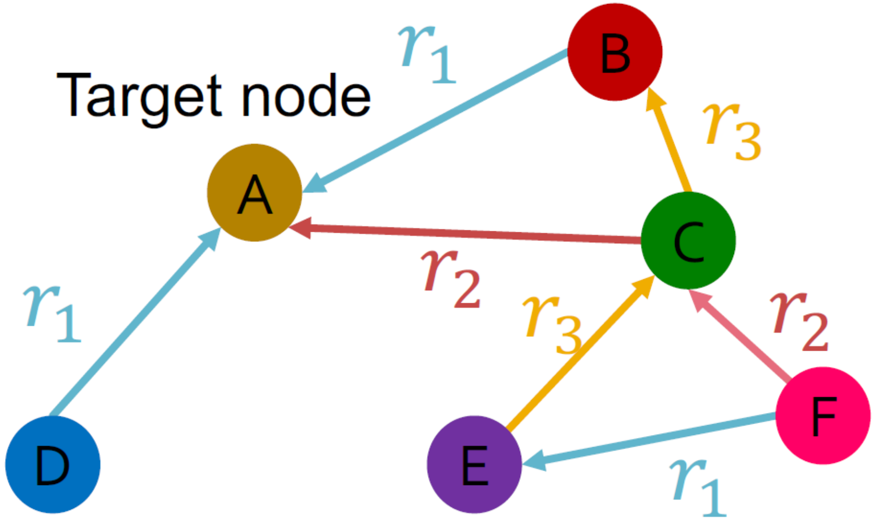
- **How to write this as Message + Aggregation?**

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



Relational GCN

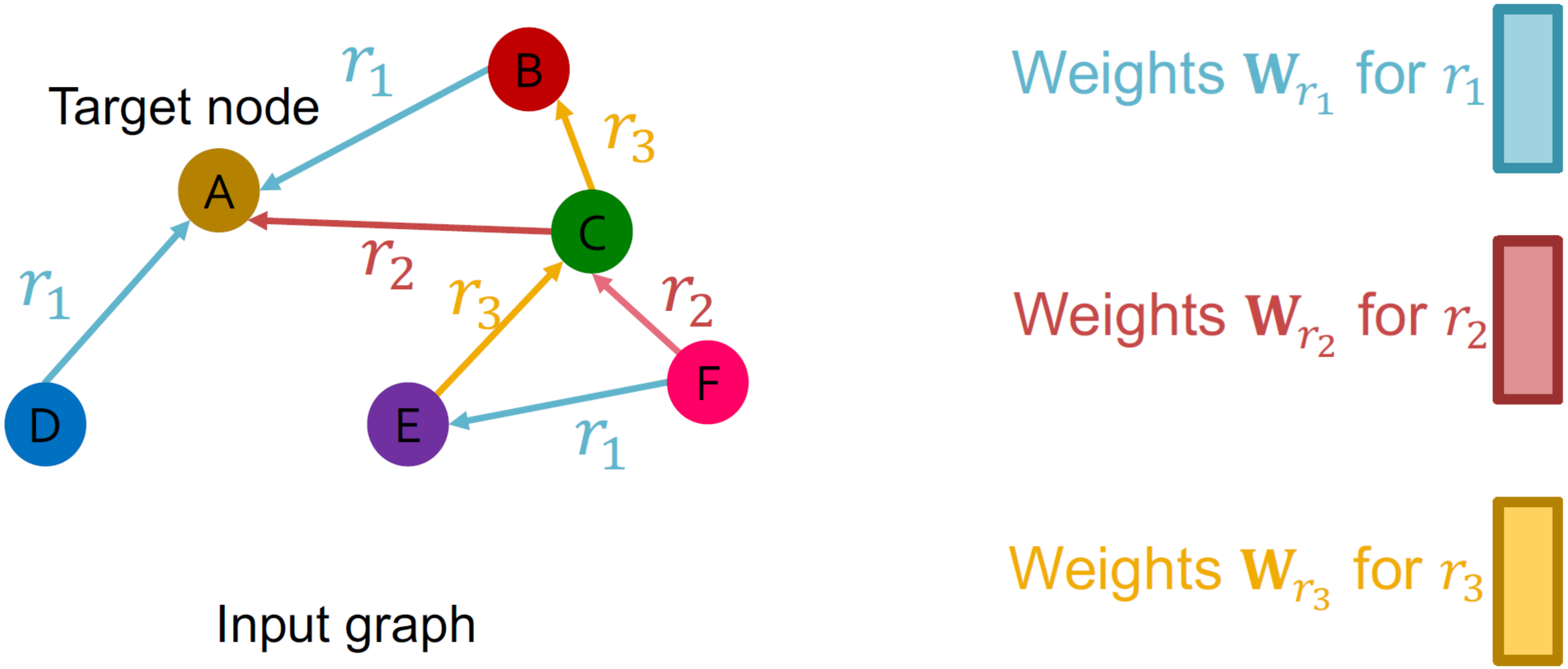
- **What if the graph has multiple relation types?**



Input graph

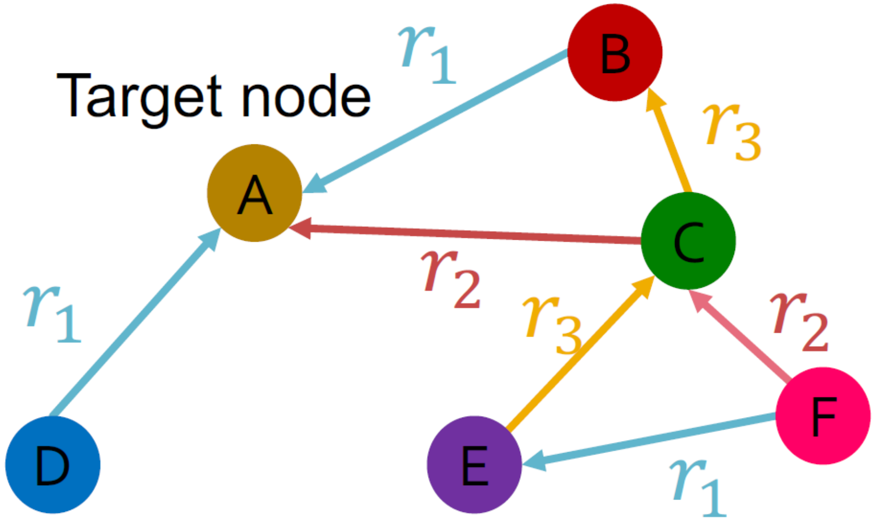
Relational GCN

- Use different neural network weights for different relation types.

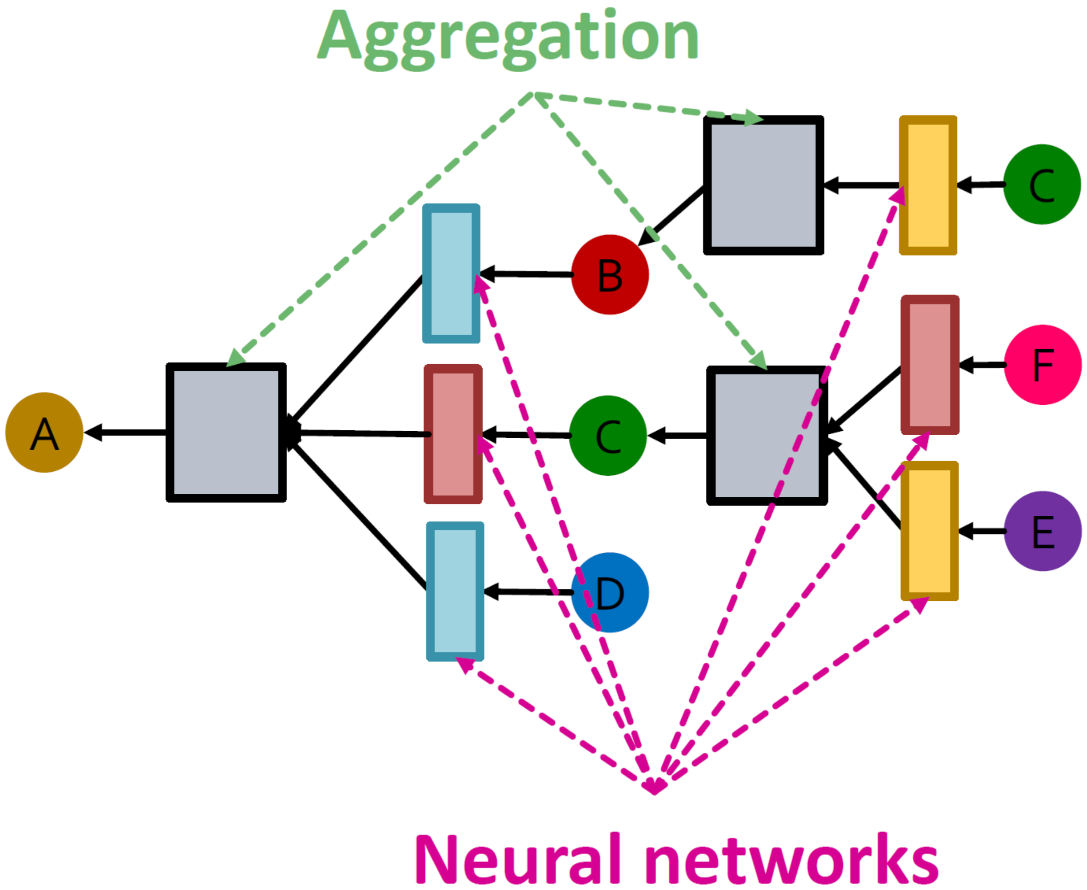


Relational GCN

- Use different neural network weights for different relation types.



Input graph



Relational GCN

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

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathrm{Sum}\left(\left\{\mathbf{m}_{u,r}^{(l)}, u \in N(v)\right\} \cup \left\{\mathbf{m}_v^{(l)}\right\}\right)\right)$$

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

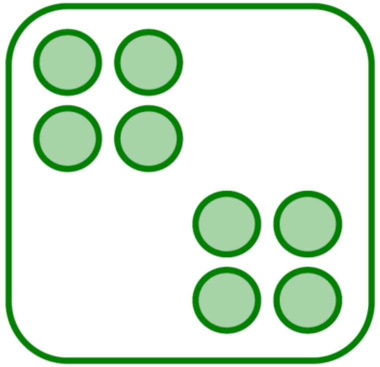
○ **Rapid # parameters growth w.r.t # relations!**
  - **Overfitting becomes an issue**

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

(1) Block Diagonal Matrices

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

$\mathbf{W}_r =$



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

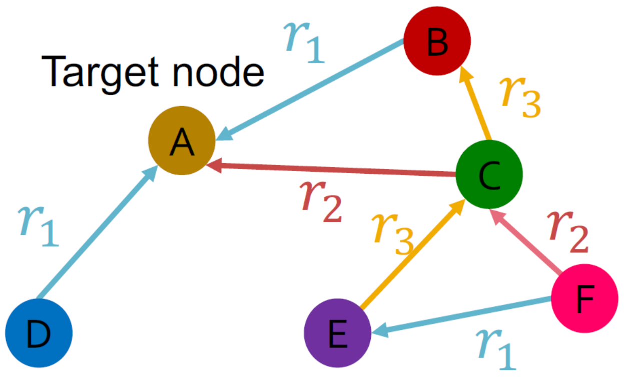
- **Key insight**: **Share weights** across different relations!
- Represent the matrix of each relation as a linear combination of basis transformations

$$\mathbf{W}_r = \sum_{b=1}^{B} a_{rb} \cdot \mathbf{V}_b$$

  - $\boldsymbol{V}_b$ is shared across relations (the basis matrices)
  - a_{rb} is the importance weight of matrix $\boldsymbol{V}_b$
- Now each relation only needs to learn $\{a_{rb}\}_{\mathrm{b}=1}^{B}$, which is B scalars.

Example: Entity/Node Classification

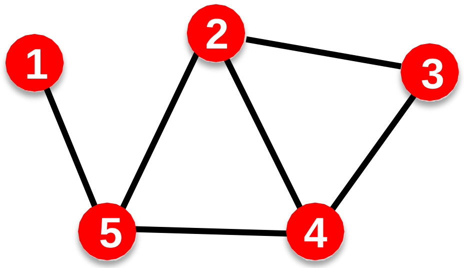
- **Goal: Predict the label of a given node**
- **RGCN** uses the representation of the final layer:
  - If we predict the class of **node A** from **k classes**.
  - Take the **final layer (prediction head)**: $\mathbf{h}_A^{(L)} \in \mathbb{R}^k$, each item in $\mathbf{h}_A^{(L)}$ represents **the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



Input graph

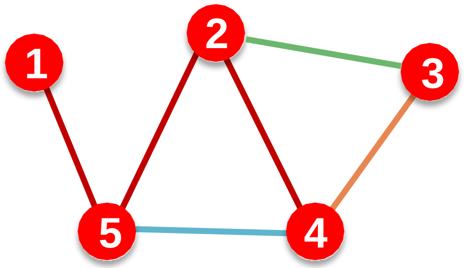
Example: Link Prediction

- **Link prediction split:**



The original graph

Split



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.

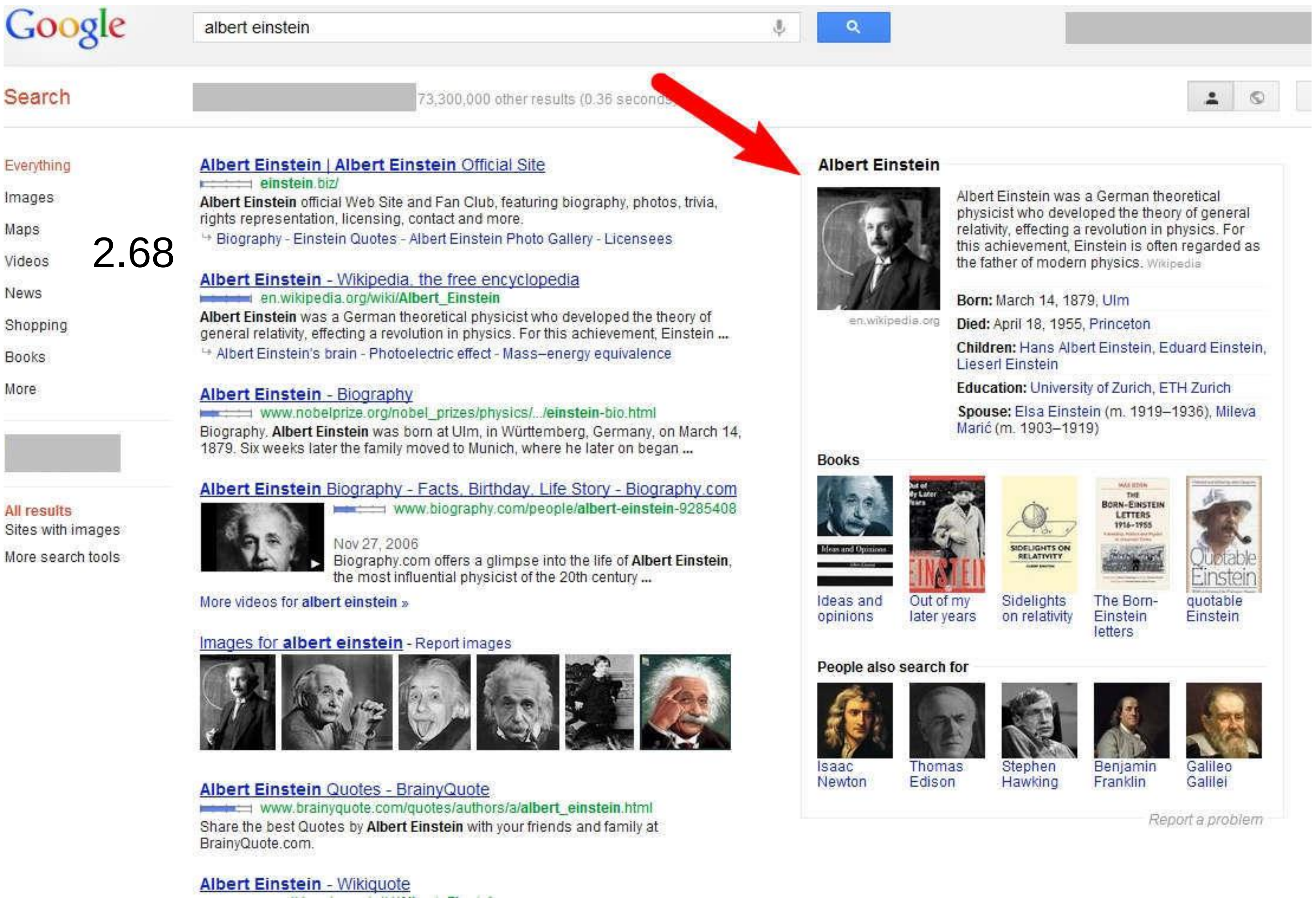
Summary of RGCN

- **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.)

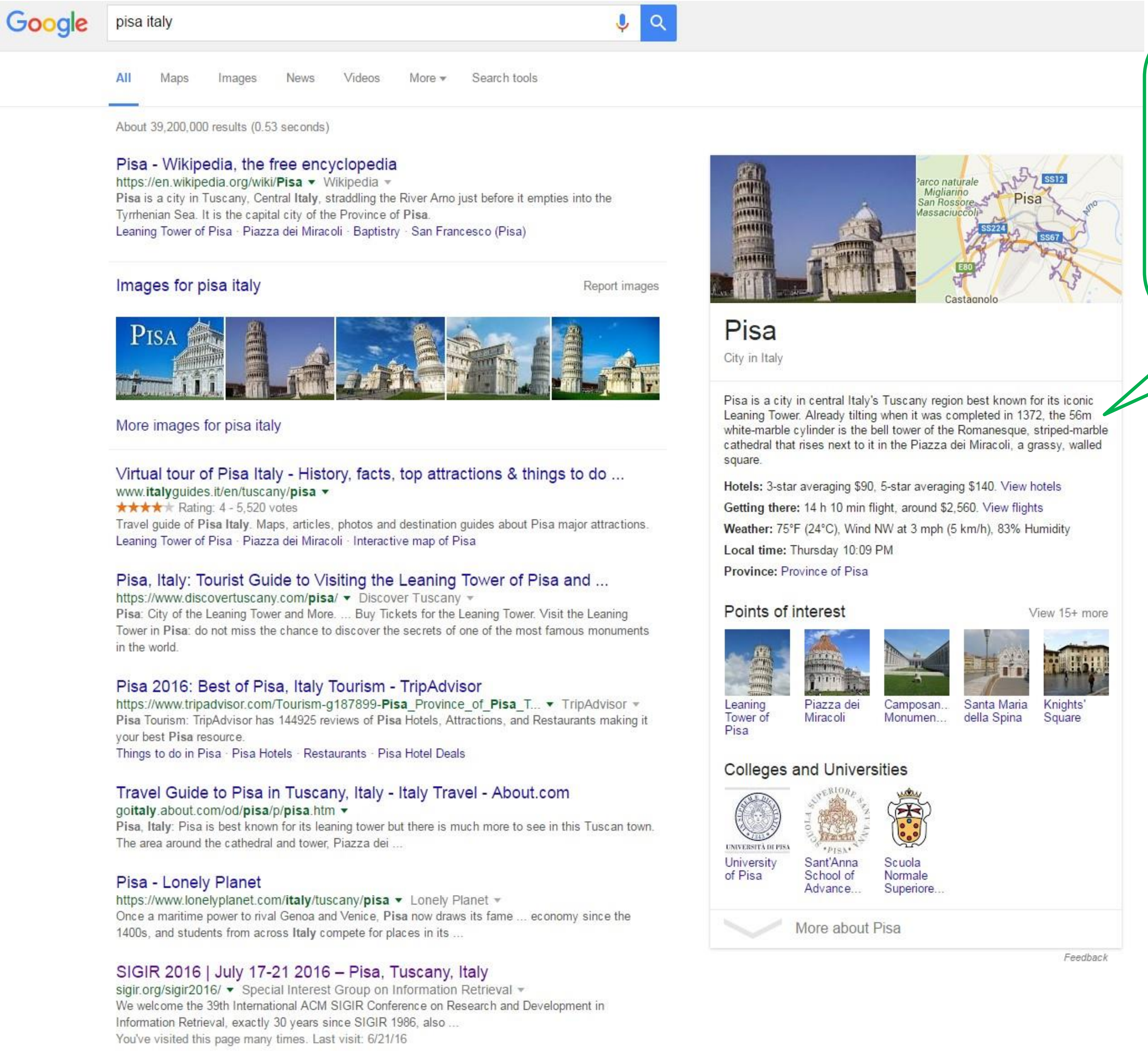
Knowledge Graphs: Completion with Embeddings

Google Knowledge Graph

- Launched in May 2012
- 570 million entities and 18 billions of relationships

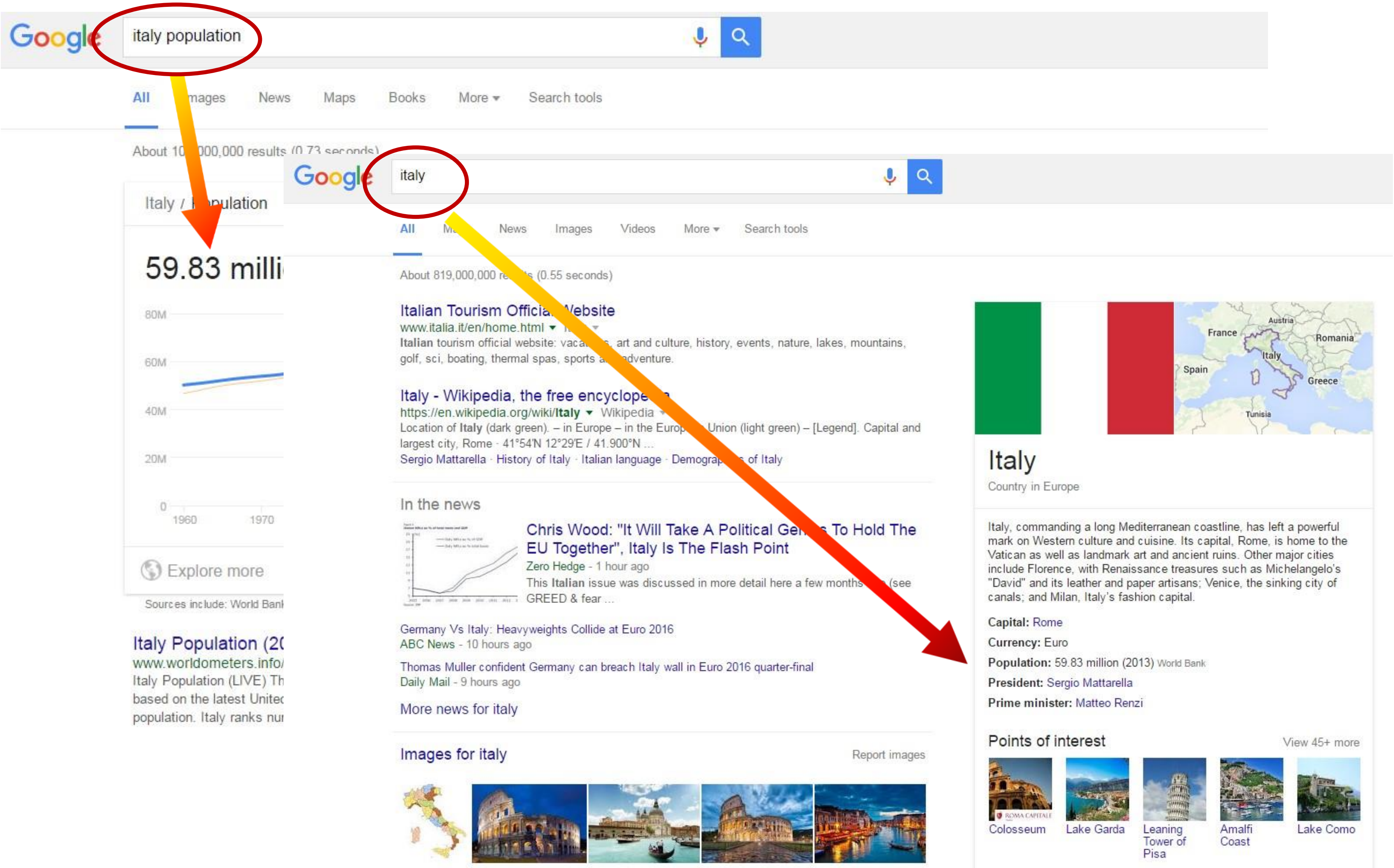


Surfacing structured results in web search

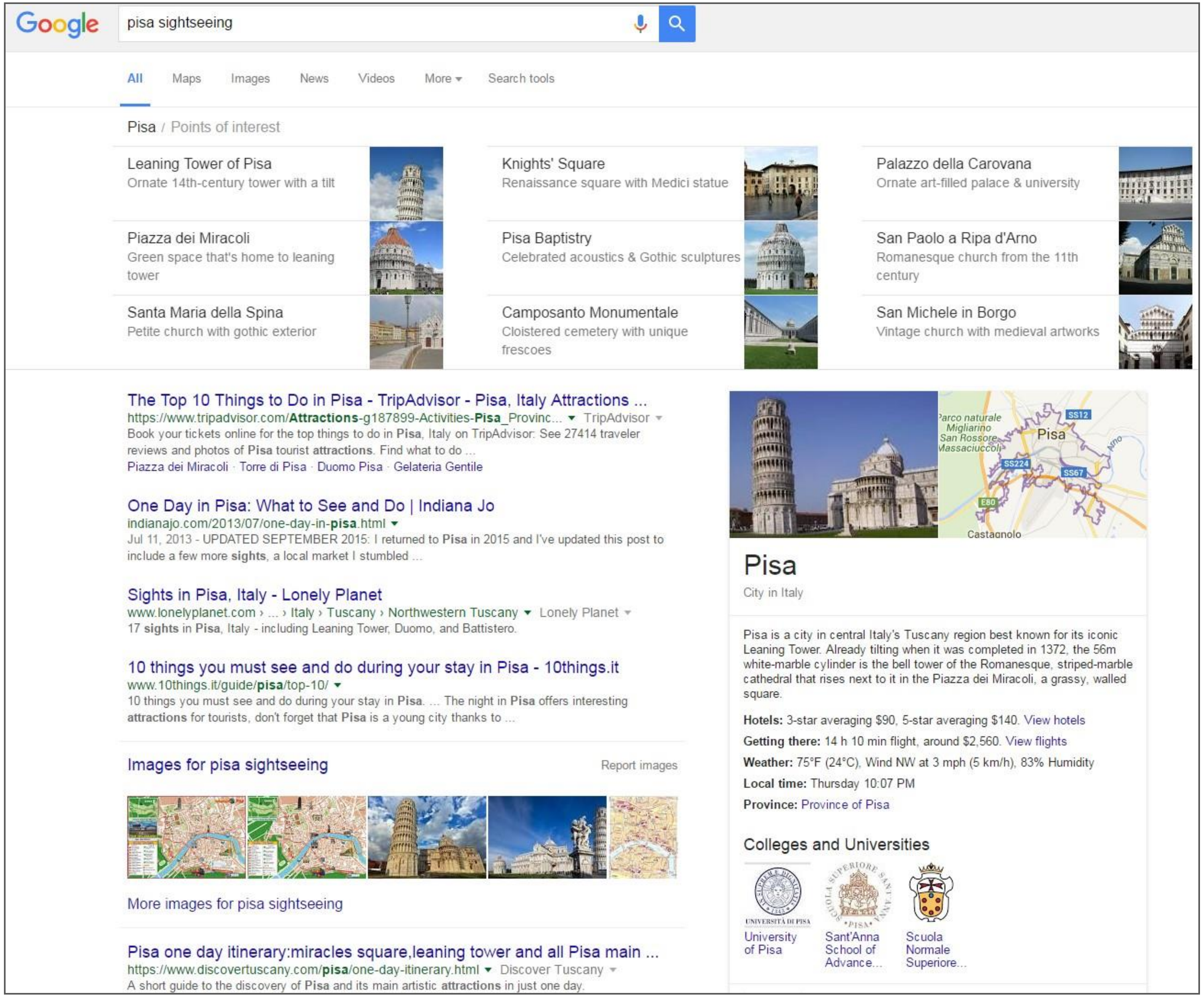


Augmenting the presentation with relevant facts

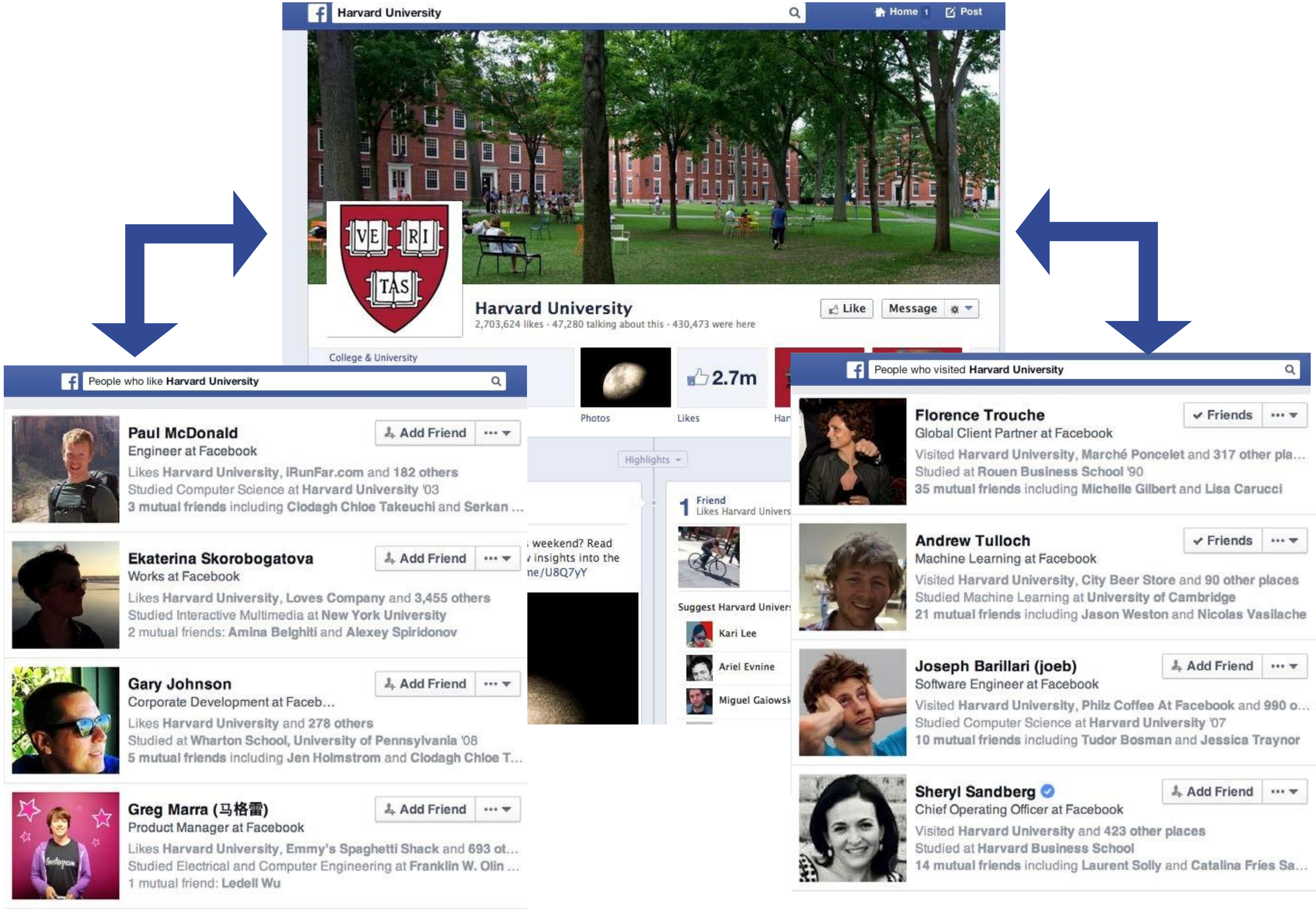
Surfacing facts proactively



Exploratory search



Connecting people, places, and things

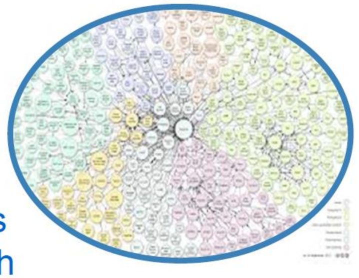


Connecting people, places and things

Structured search within the graph

Knowledge Graph Summary

- WordNet
  - Expert annotation
  - Linguistic
- Cyc
  - Expert annotation
  - Formalized logical reasoning
- ConceptNet (MIT)
  - Crowdsourcing
  - Context-based inferences over real-world texts
- KnowItAll (UW), NELL (CMU)
  - Information Extraction
  - Triplets (entities and relations)
- Google Knowledge Graph
  - Combination
  - Triplets (entities and relations)
- Amazon Product Graph
  - More products and attributes



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**

Freebase

93.8% of persons from Freebase have no place of birth and 78.5% have no nationality!

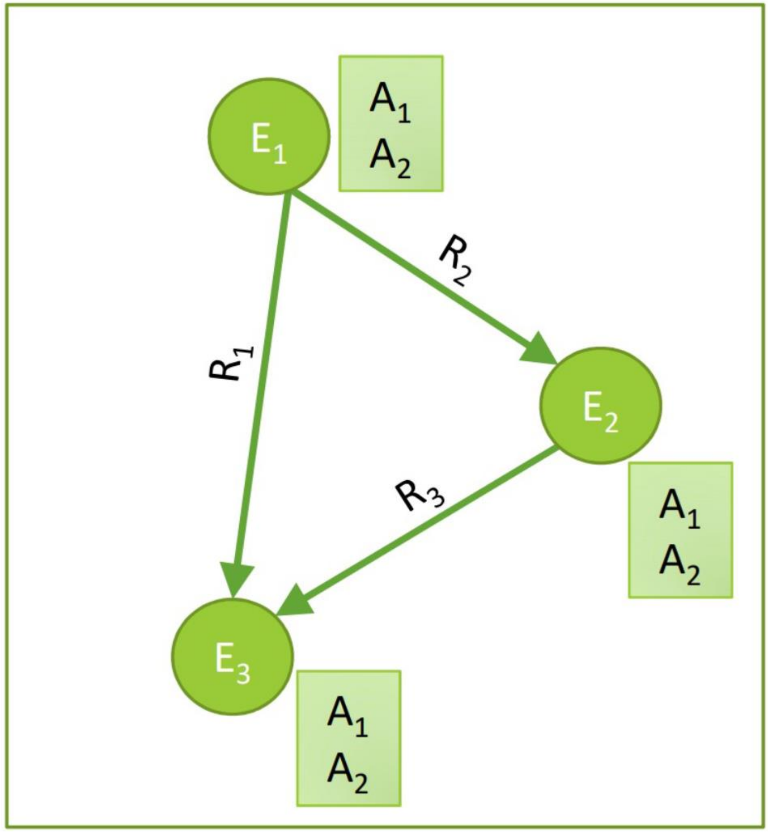
- **Datasets:** FB15k/FB15k-237
  - A complete subset of Freebase, used by researchers to learn KG models

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.
2Min, 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.

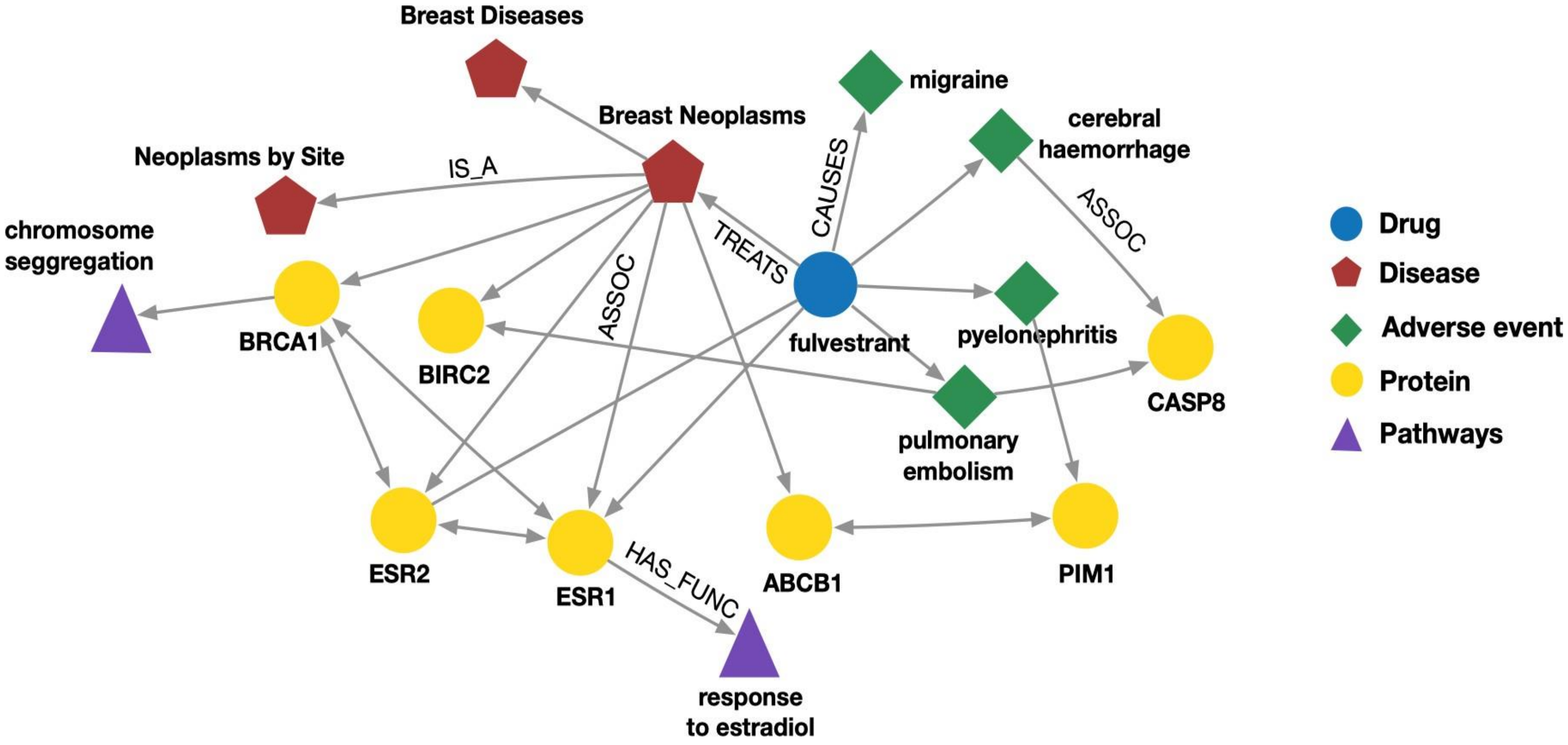
Knowledge Graph Representation

- 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
- Some entities also have **attributes**
- **KG is an example of a heterogeneous graph**



Bio Knowledge Graphs

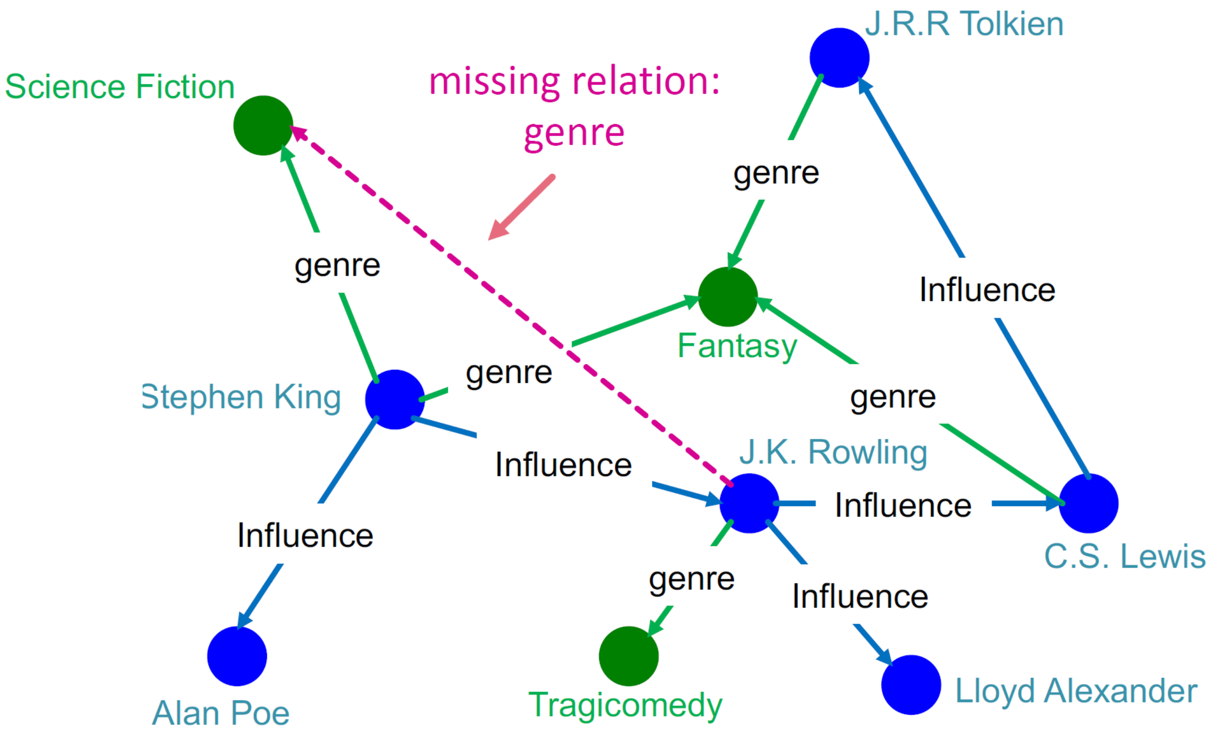
- **Node types**: drug, disease, adverse event, protein, pathways
- **Relation types**: has_func, causes, assoc, treats, is_a



KG Completion Task

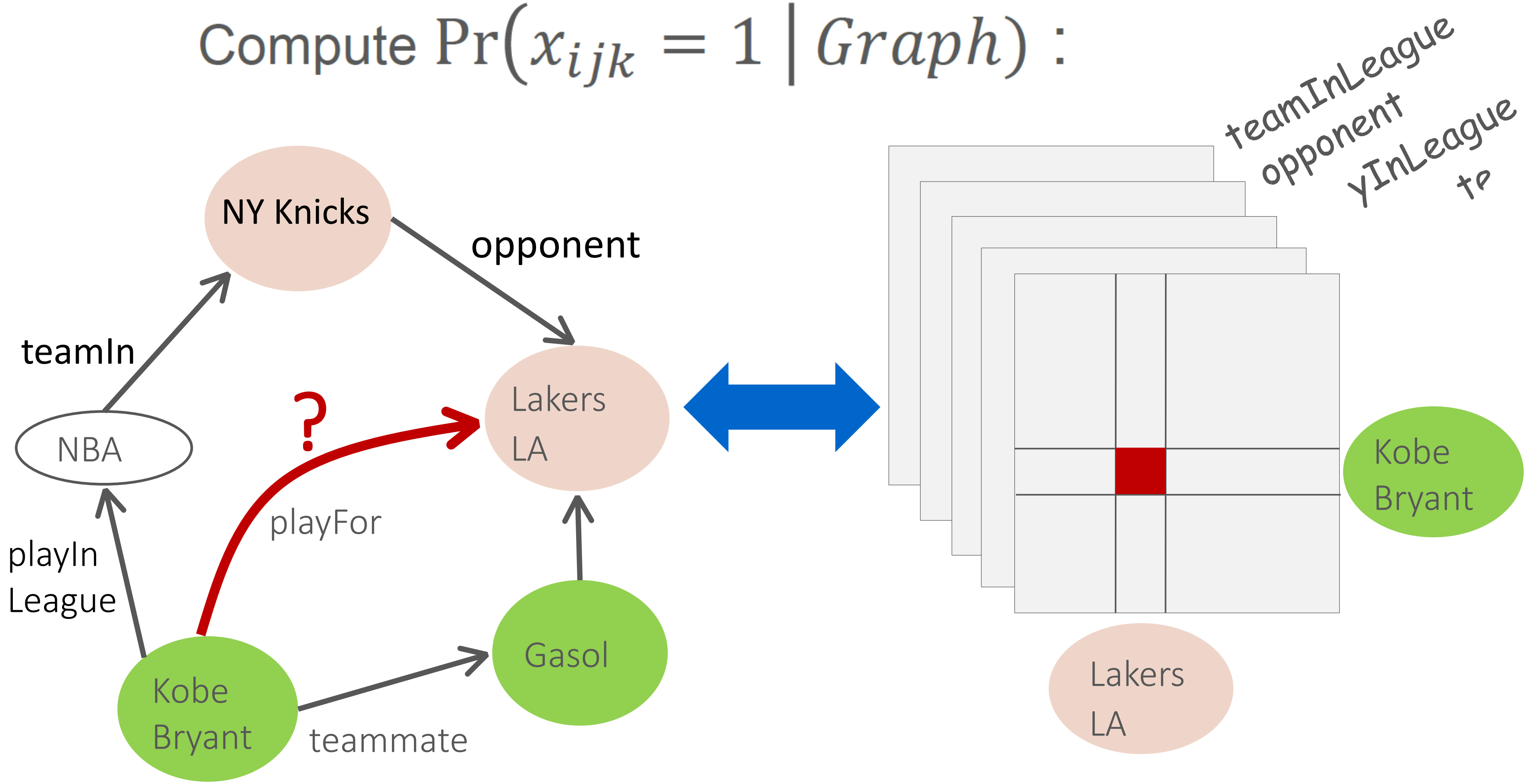
- Given an enormous KG, can we complete the KG?
- For a given (**head**, **relation**), we predict missing **tails**.
  - (Note this is slightly different from link prediction task)

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



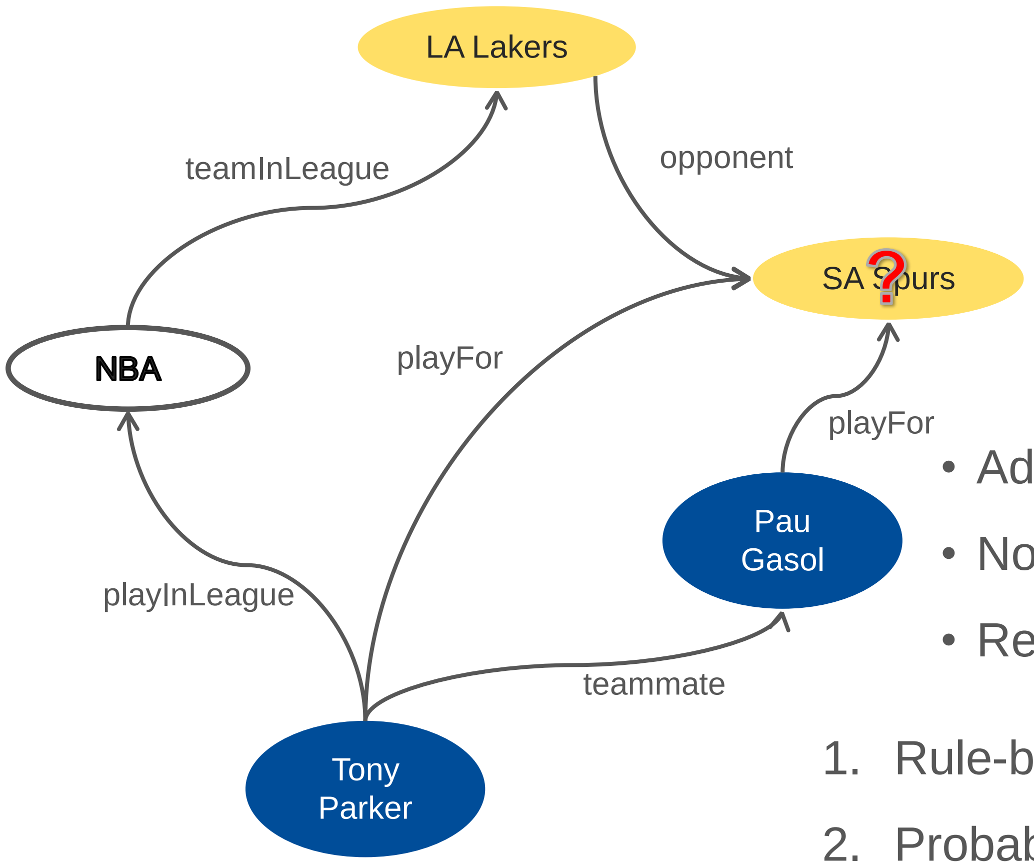
Knowledge Graph Completion as Link Prediction

Compute $\Pr(x_{ijk} = 1 \mid Graph)$:



"A Review of Relational Machine Learning for Knowledge Graphs" Nickel, Murphy, Tresp, and Gabrilovich, Proceedings of the IEEE, 104(1):11-33, 2016

Link Prediction

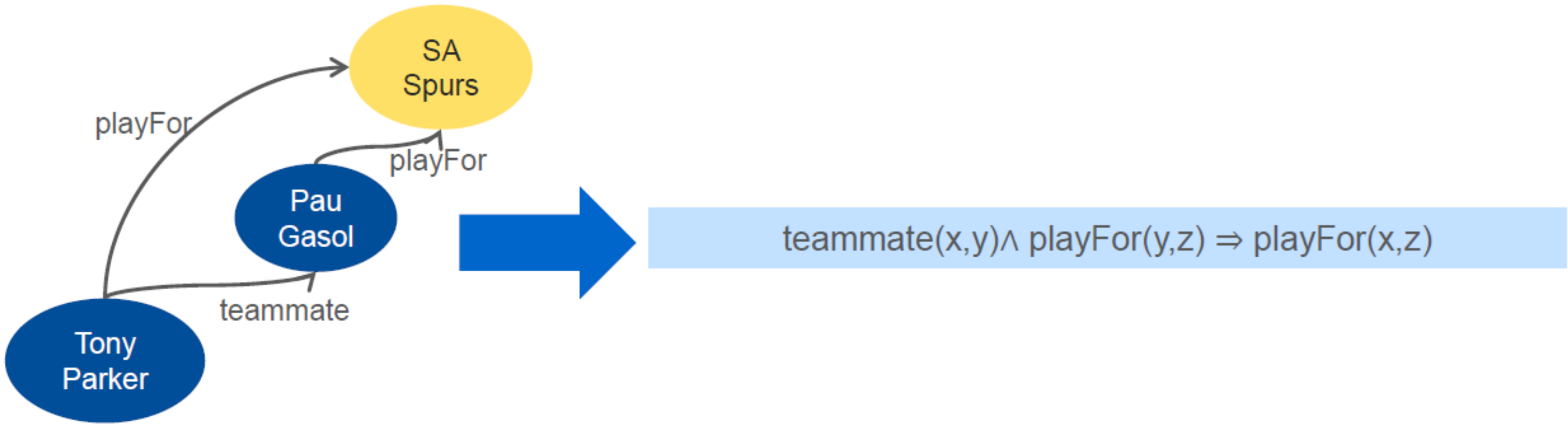


- Add knowledge from existing graph
- No external source
- Reasoning within the graph

1. Rule-based methods
2. Probabilistic models
3. Factorization models
4. Embedding models

First Order Inductive Learner

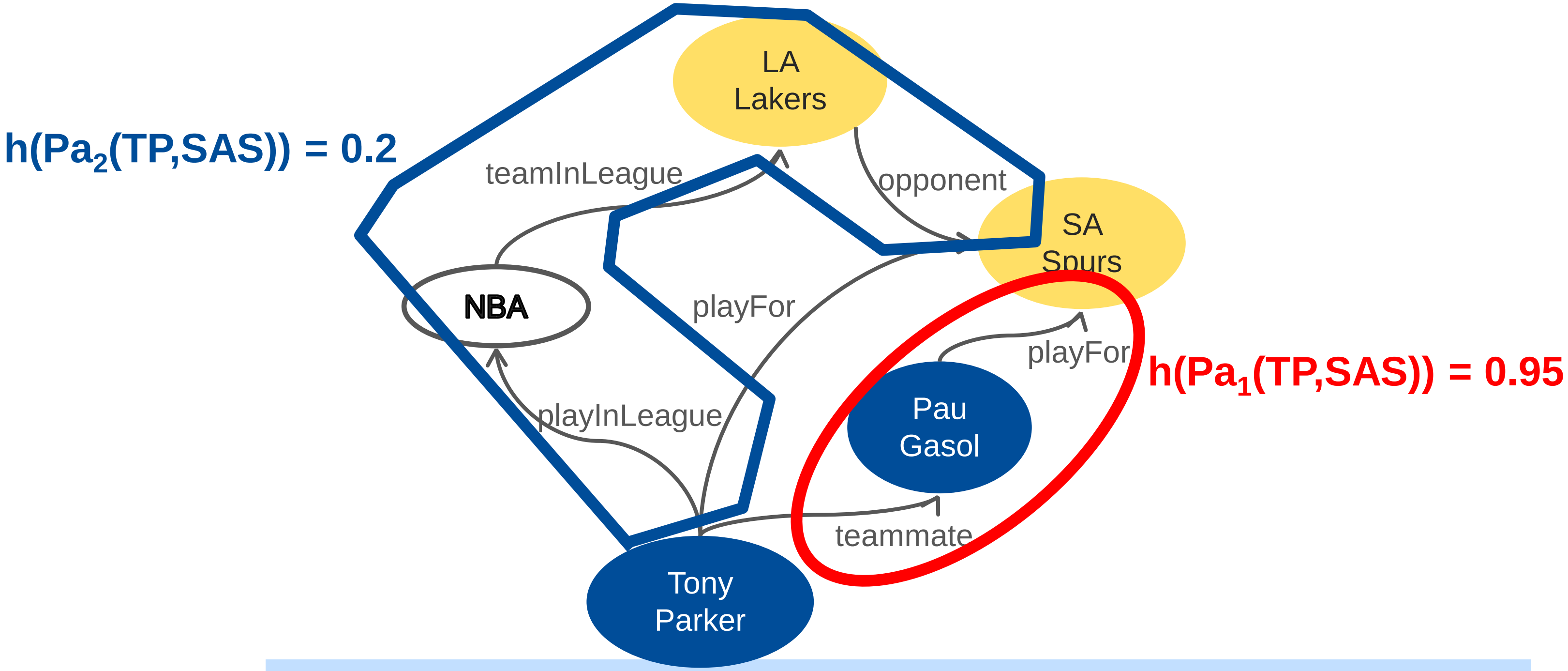
- FOIL learns function-free Horn clauses:
  - given positive negative examples of a concept
  - a set of background-knowledge predicates
  - FOIL inductively generates a logical rule for the concept that cover all + and no –



- Computationally expensive: huge search space large, costly Horn clauses
- Must add constraints → high precision but low recall
- Inductive Logic Programming: deterministic and potentially problematic

Path Ranking Algorithm [Lao et al., 11]

- Random walks on the graph are used to sample paths
- Paths are weighted with probability of reaching target from source
- Paths are used as ranking experts in a scoring function



$$S(TP, playFor, SAS) = \sum_{i \in paths} \theta^{i}_{playFor} h(pa_i(TP, SAS))$$

Link Prediction with Scoring Functions

- A scoring function alone does not grant a decision $S(TP, playFor, SAS) > \theta$
- **Thresholding**: determine a threshold ϑ

  (*TP, playFor, SAS*) is *True* iff

- **Ranking**
  - The most likely relation between Kobe Bryant and LA Lakers is:

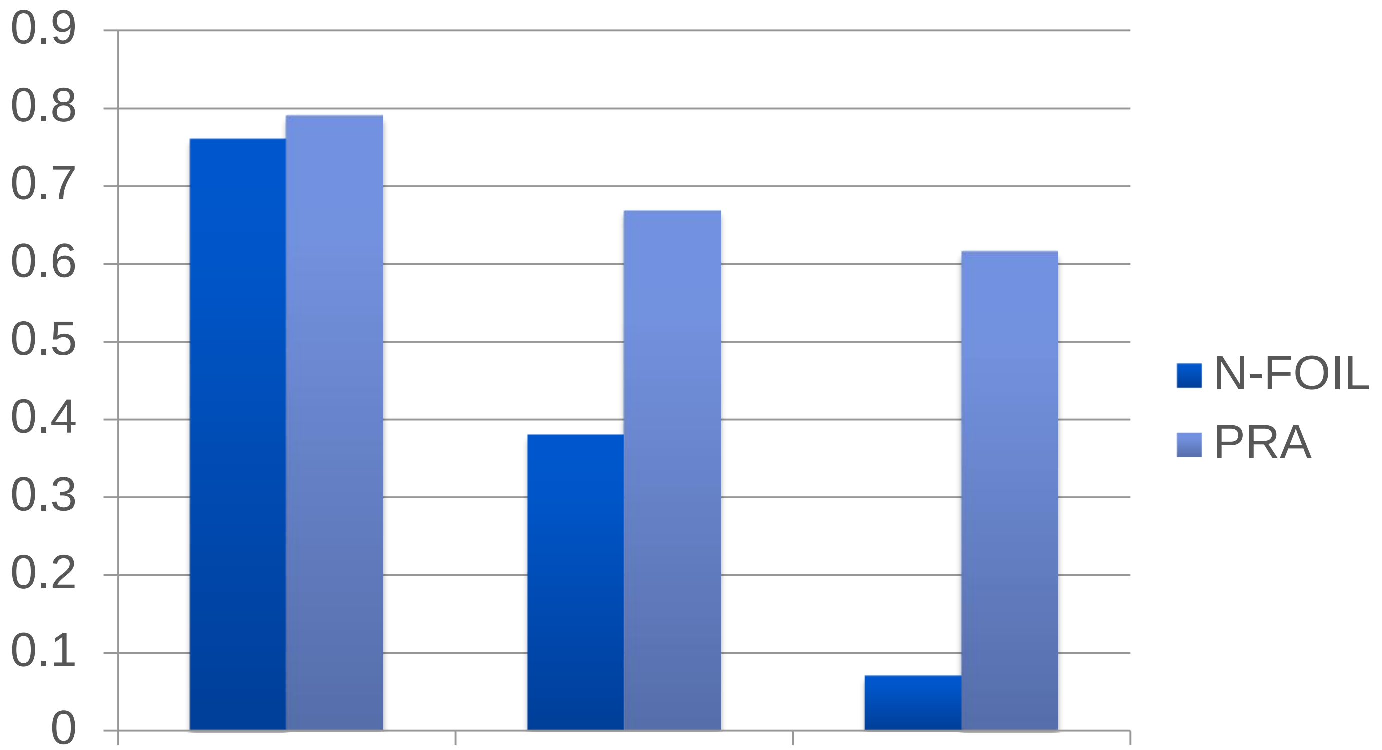
    $$rel = \text{argmax}_{r' \in rels} S(TP, r', SAS)$$

  - The most likely team for Kobe Bryant is:

    $$obj = \text{argmax}_{e' \in ents} S(TP, playFor, e')$$

- As prior for extraction models (cf. Knowledge Vault [Dong et al., 2014])
- No calibration of scores like probabilities

Random Walks Boost Precision

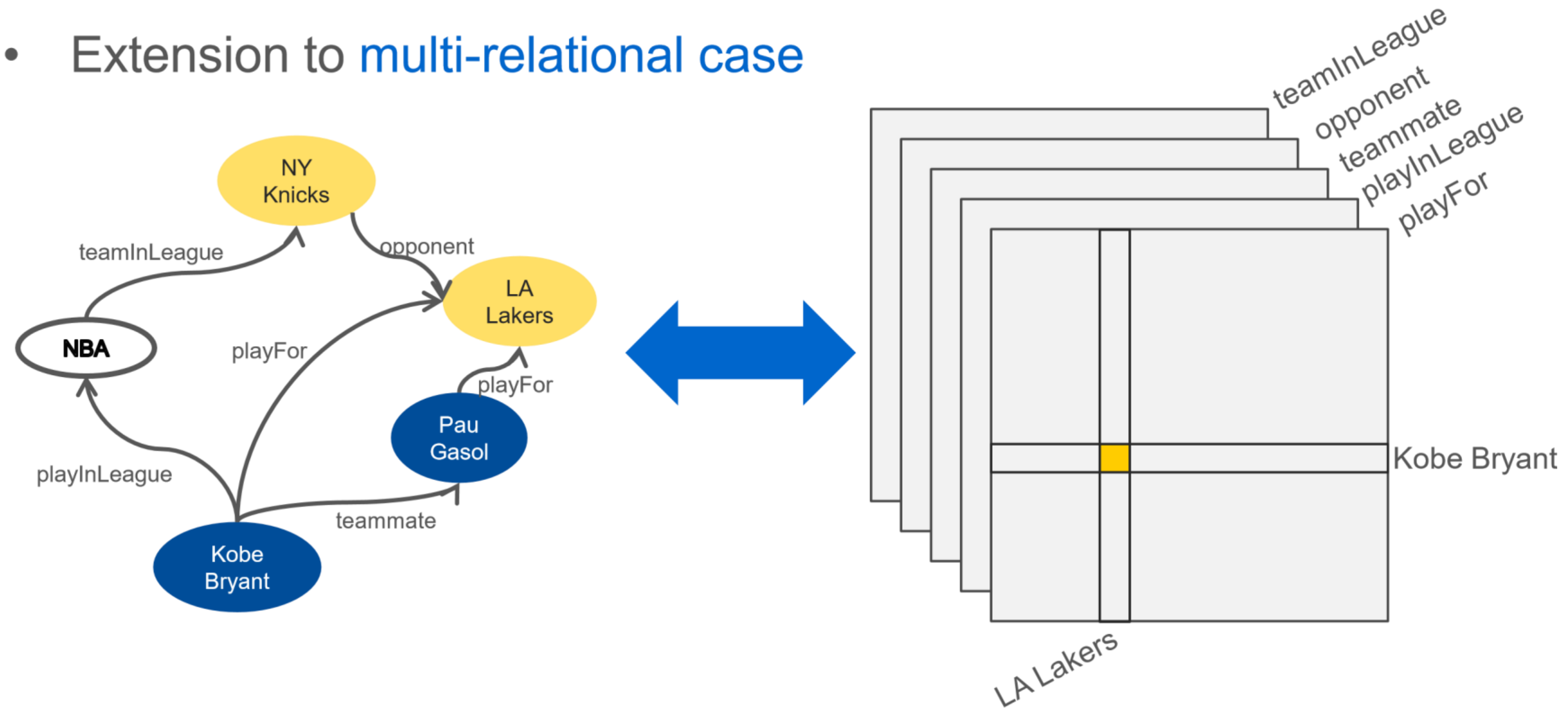


Precision of generalized facts for three cutoffs [Lao et al., 2011]

Factorization methods

- Matrix factorization is successful: collaborative filtering, recommendation, etc.

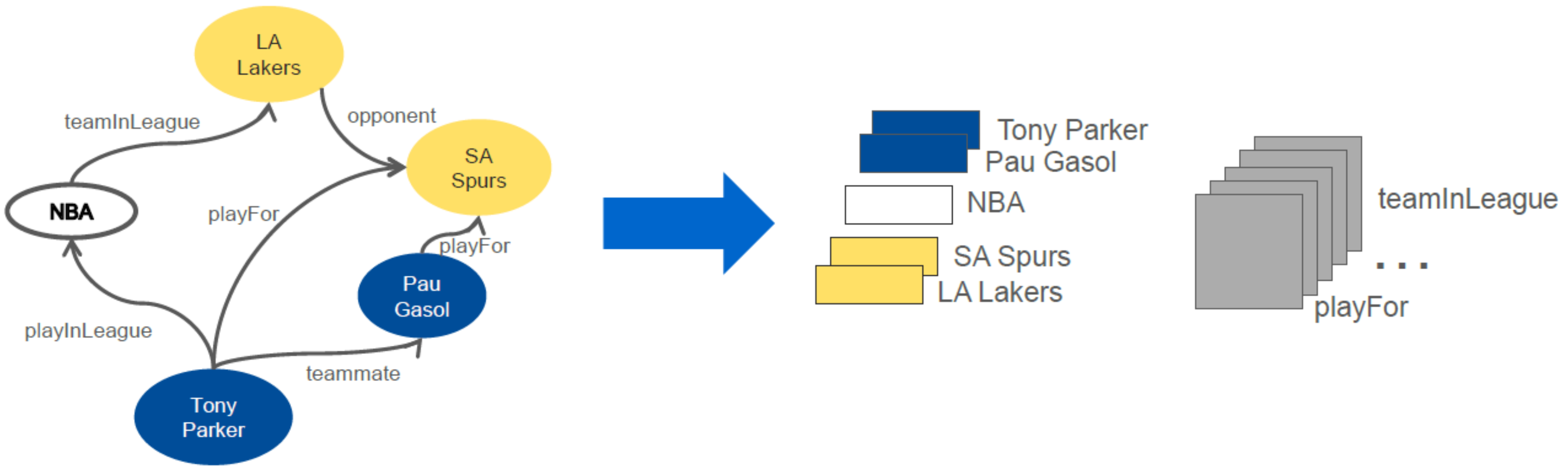
  - Extension to multi-relational case



  - Collective matrix factorization or tensor factorization

Embedding models

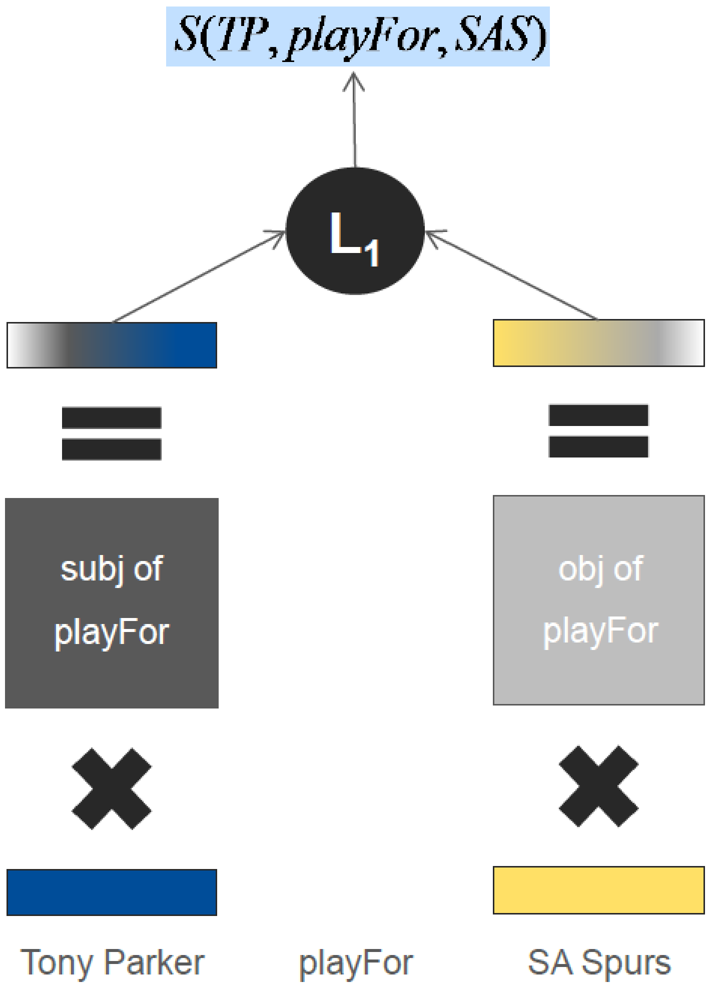
- Related to Deep Learning methods
- Entities are vectors (low-dimensional sparse)
- Relation types are operators on these vectors



Embeddings trained to define a **similarity score** on triples such that

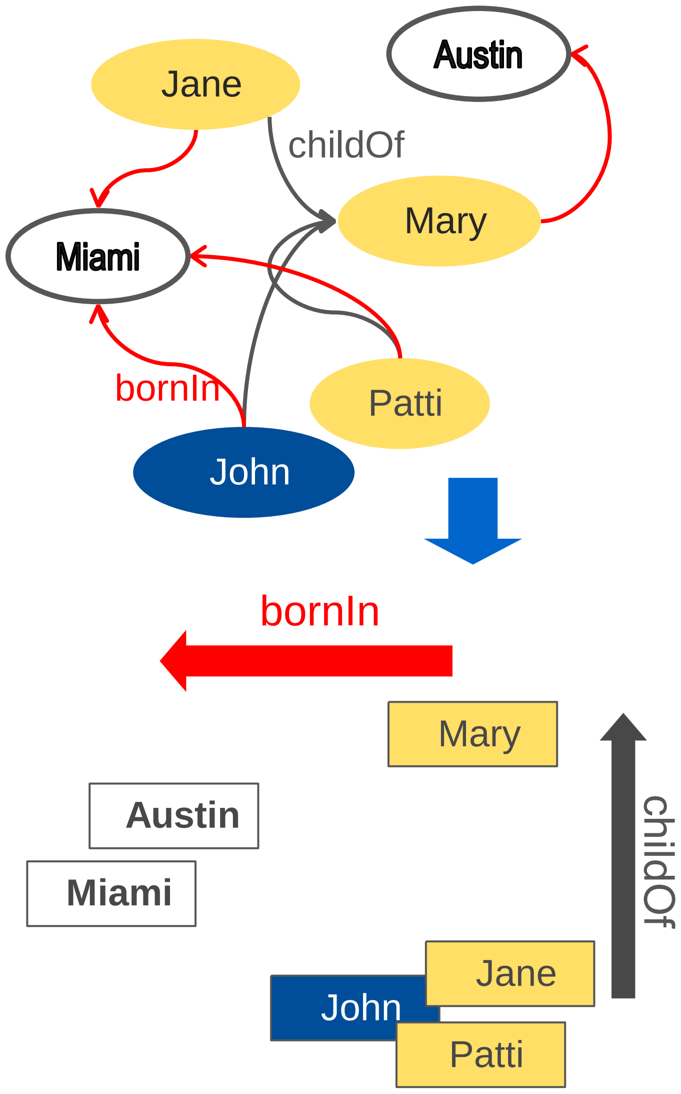
$$S(KB, playFor, LAL) > S(KB, playFor, NYK)$$

Structured Embeddings [Bordes et al., 11]



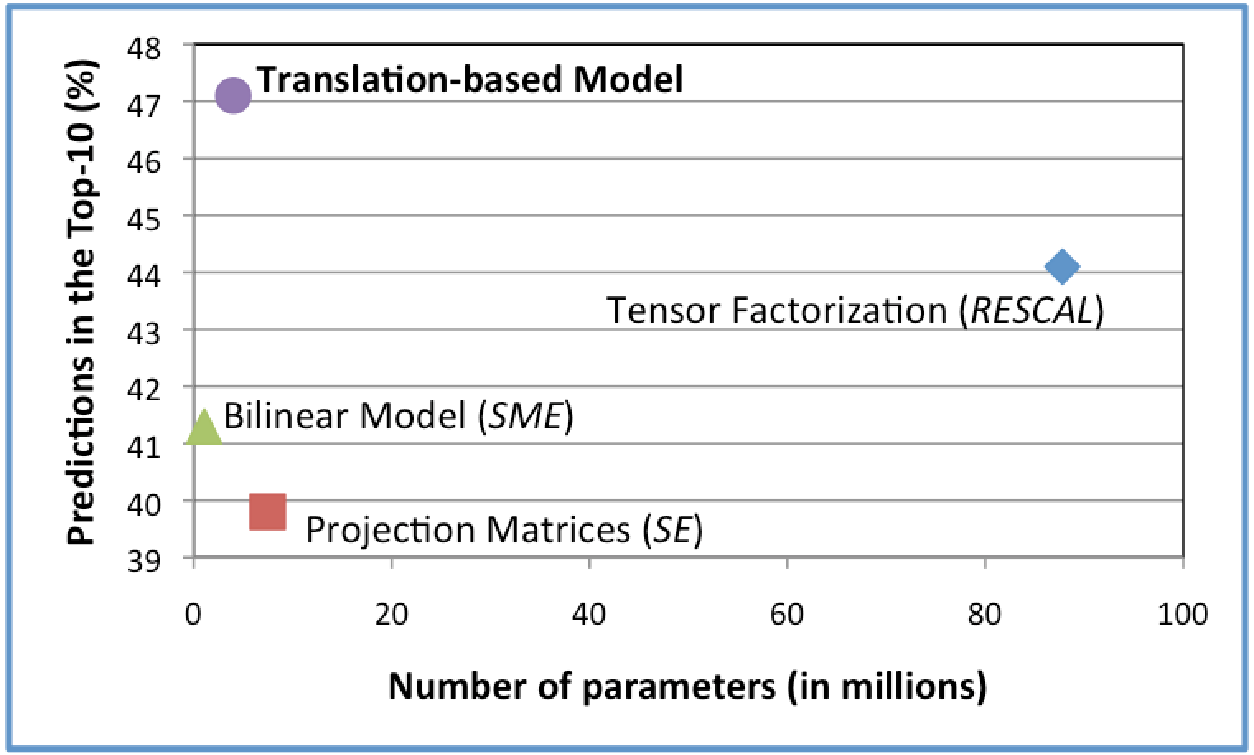
$$S(TP, playFor, SAS) = -\left\| M^{sub}_{playFor} e_{TP} - M^{obj}_{playFor} e_{SAS} \right\|_1$$

Translating Embeddings [Bordes et al. 13]



$$S(john, bornIn, miami) = -\left\| e_{john} + e_{bornIn} - e_{miami} \right\|_2$$

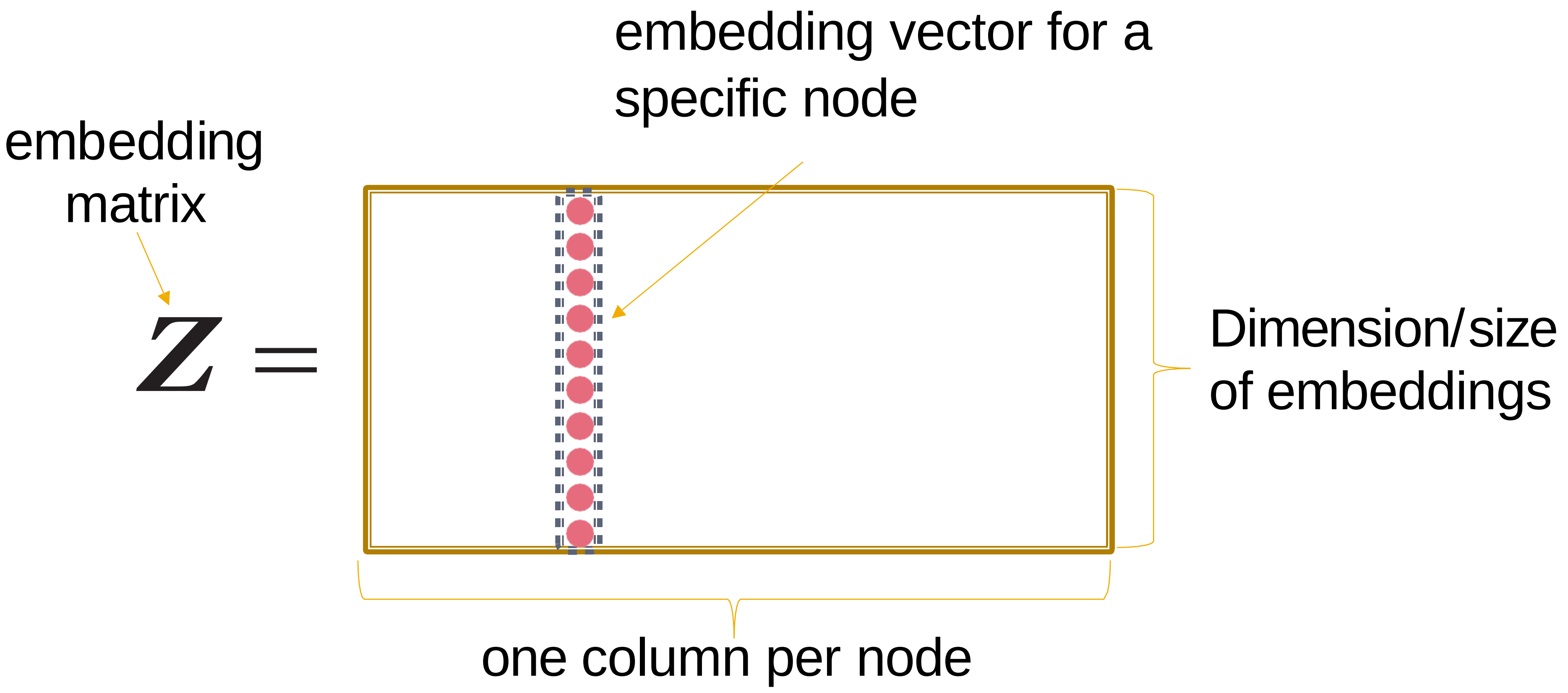
The Simpler, The Better



Ranking object entities on a subset of Freebase [Bordes et al. 13]

Recap: Shallow Encoding

- Simplest encoding approach: **encoder is just an embedding-lookup**



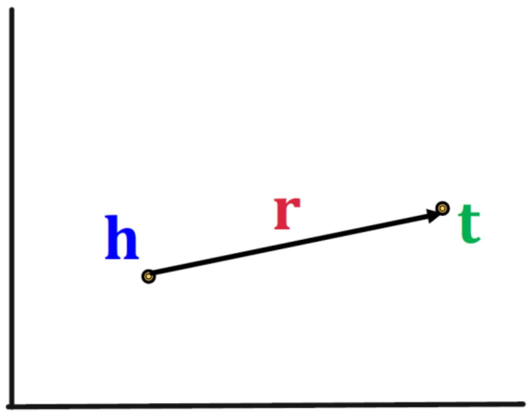
Formal KG Representation

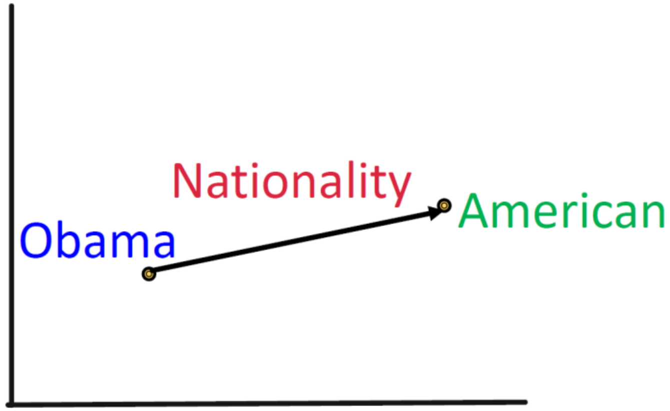
- Edges in KG are represented as **triples** (h, r, t)
  - head (h) has relation (r) with tail (t)
- **Key Idea:**
  - Model entities and relations in the embedding/vector space $\mathbb{R}^{\mathrm{d}}$.
  - Associate entities and relations with **shallow embeddings**
  - **Note we do not learn a GNN 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.

TransE

embedding vectors will appear in boldface

- **Translation Intuition**:
- For a triple (h, r, t), $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^{\mathrm{d}}$,
  - $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ if the given fact is true
  - $\mathbf{h} + \mathbf{r} \neq \mathbf{t}$ else
  - $f_{\mathrm{r}}(h, t) = -\mathrm{d}_{\mathrm{r}}(h, t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||$
- **Loss function**: $L(h, r, t) = \max(0, \gamma + f_{r,neg} - f_{r,pos})$
  $= \max(0, \gamma + d_{r,pos} - d_{r,neg})$
- In learning, we minimize the loss function for all pairs of triples $\min L(h, r, t)$





TransE 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 $e \in E$

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

Entities and relations are initialized uniformly, and normalized

Negative sampling with triplet 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

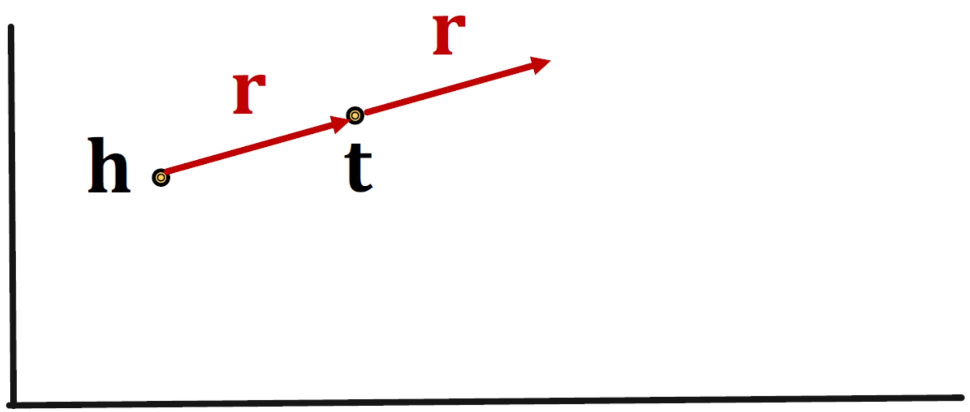
- **Relations in a heterogeneous KG have different properties**
  - Example
    - **Symmetry:** If the edge $(h, \text{"Roommate"}, t)$ exists in KG, then the edge $(t, \text{"Roommate"}, h)$ should also exist.
    - **Inverse relation:** If the edge $(h, \text{"Advisor"}, t)$ exists in KG, then the 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:**
  - **Example**: $r(h,t) \Rightarrow r(t,h) \;\; (r(h,t) \Rightarrow \neg r(t,h)) \;\; \forall h,t$
    - Symmetric: Family, Roommate
    - Antisymmetric: Hypernym
- **Inverse Relations:** $r_2(h,t) \Rightarrow r_1(t,h)$
  - **Example** : (Advisor, Advisee)
- **Composition (Transitive) Relations:** $r_1(x,y) \wedge r_2(y,z) \Rightarrow r_3(x,z) \;\; \forall x,y,z$
  - **Example**: My mother's husband is my father.
- **1-to-N relations:** $r(h,t_1), r(h,t_2), \dots, r(h,t_n)$ are all True.
  - **Example**: r is "StudentsOf"

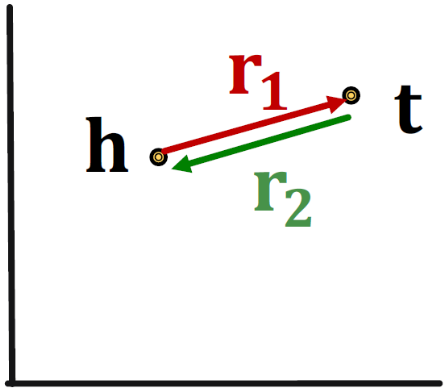
Antisymmetric Relations in TransE

- **Antisymmetric Relations:**
  - $r(h, t) \Rightarrow \neg r(t, h)\ \forall h, t$
  - **Example**: Hypernym
- **TransE** can model antisymmetric relations ✓
  - $\mathbf{h} + \mathbf{r} = \mathbf{t}$, but $\mathbf{t} + \mathbf{r} \neq \mathbf{h}$



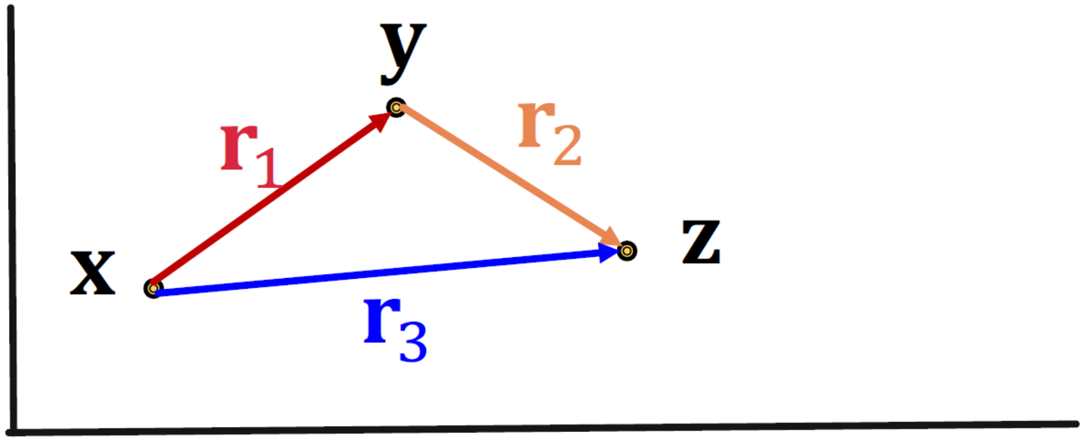
Inverse Relations in TransE

- **Inverse Relations:**
  - $r_1(h, t) \Rightarrow r_2(t, h)$
  - **Example** : (Advisor, Advisee)
- **TransE** can model inverse relations ✓
  - $\mathbf{h} + \mathbf{r}_2 = \mathbf{t}$, we can set $\mathbf{r}_1 = -\mathbf{r}_2$



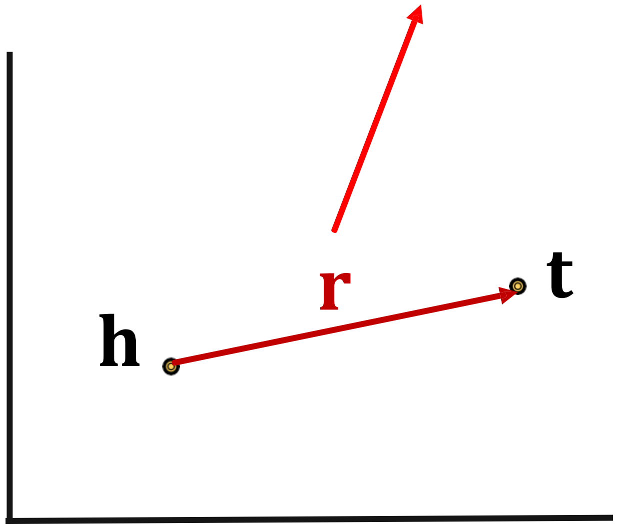
Composition in TransE

- **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 ✓
  - $\mathbf{r}_1 = \mathbf{r}_2 + \mathbf{r}_3$



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.

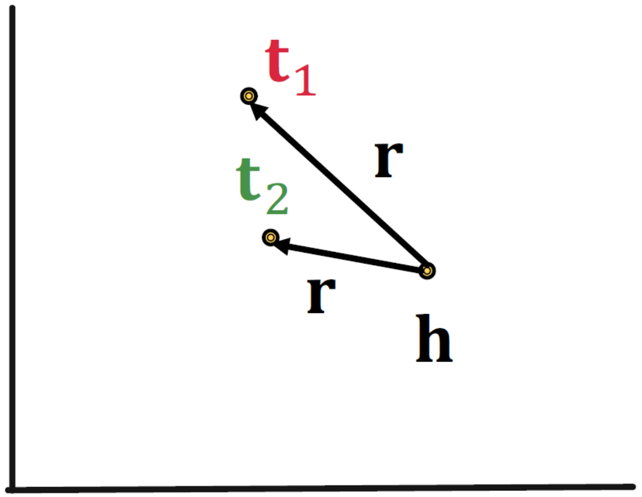
Limitation: 1-to-N Relations

- **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 $\mathbf{M}_\mathrm{r} \in \mathbb{R}^{k \times d}$ as the projection matrix.

TransR

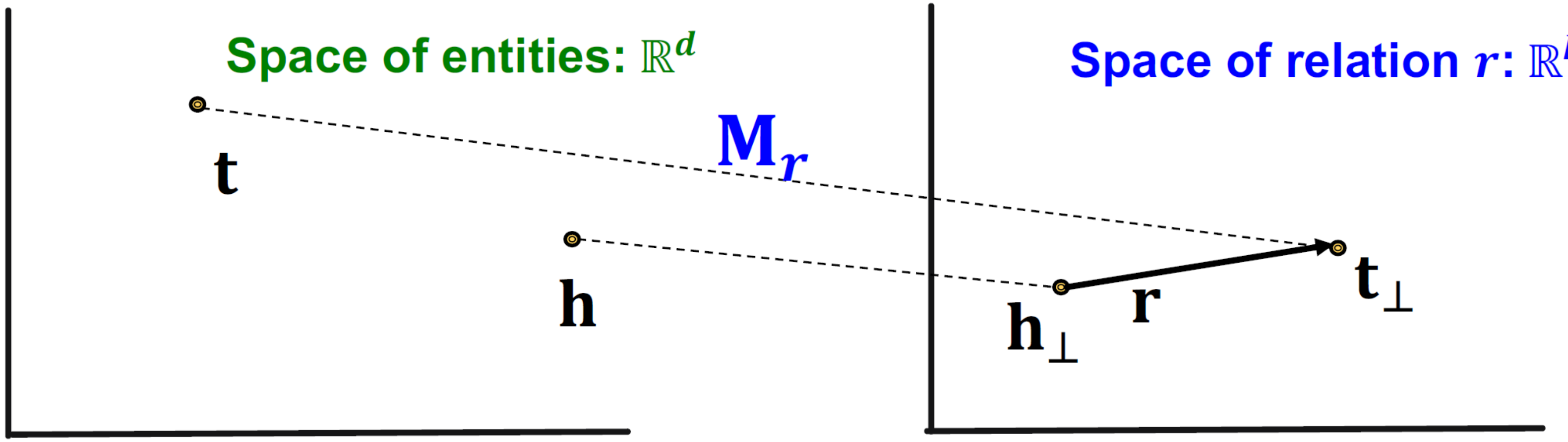
- **TransR**: model **entities** as vectors in the entity space $\mathbb{R}^d$ and model each **relation** as vector in relation space $\mathbf{r} \in \mathbb{R}^k$ with $\mathbf{M}_r \in \mathbb{R}^{k \times d}$ as the projection matrix.

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

- Score function:

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

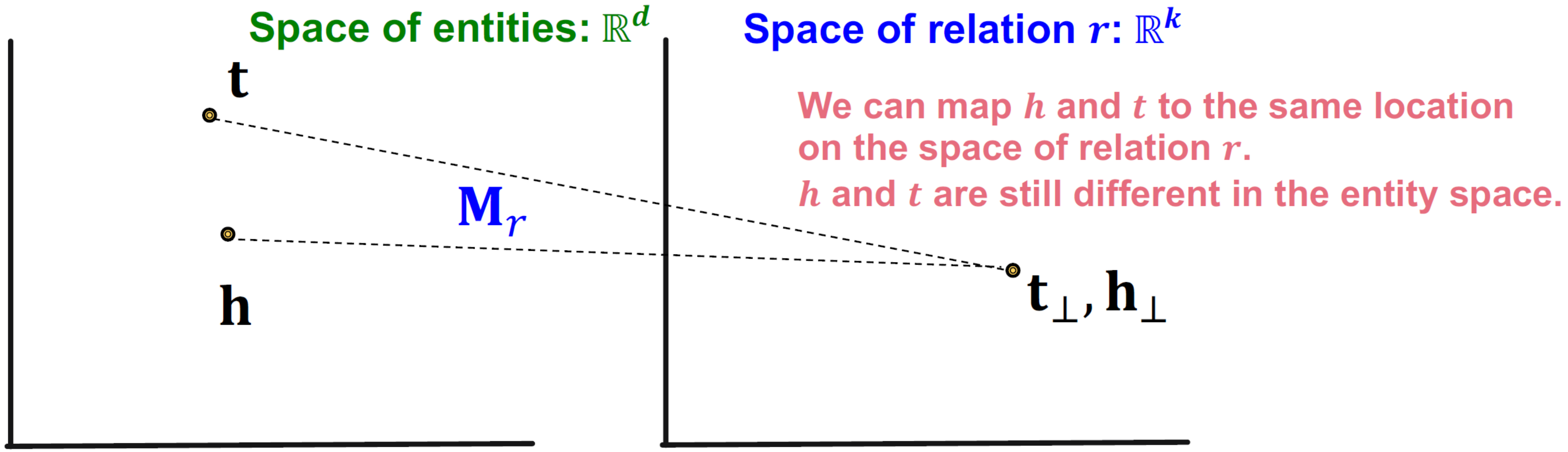
$$f_r(h,t) = -||\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp||$$



Symmetric Relations in TransR

- **Symmetric Relations:**
  - $r(h, t) \Rightarrow r(t, h) \; \forall h, t$
- **Example**: Family, Roommate
- **TransR** can model symmetric relations

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

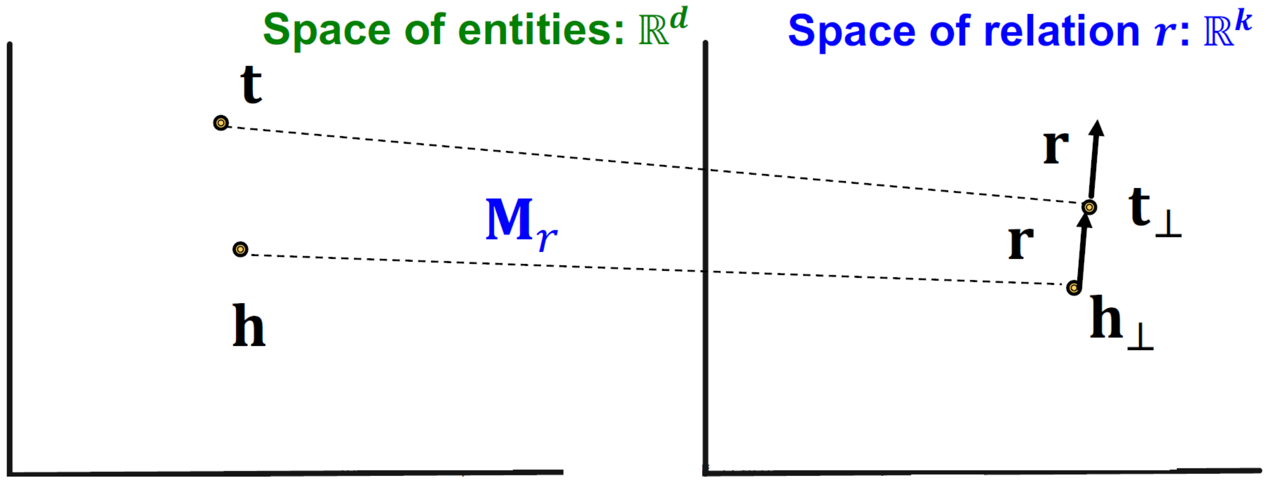


Antisymmetric Relations in TransR

- **Antisymmetric Relations:**
    - $r(h,t) \Rightarrow \neg r(t,h)\ \forall h,t$
    - **Example**: Hypernym
- **TransR** can model antisymmetric relations

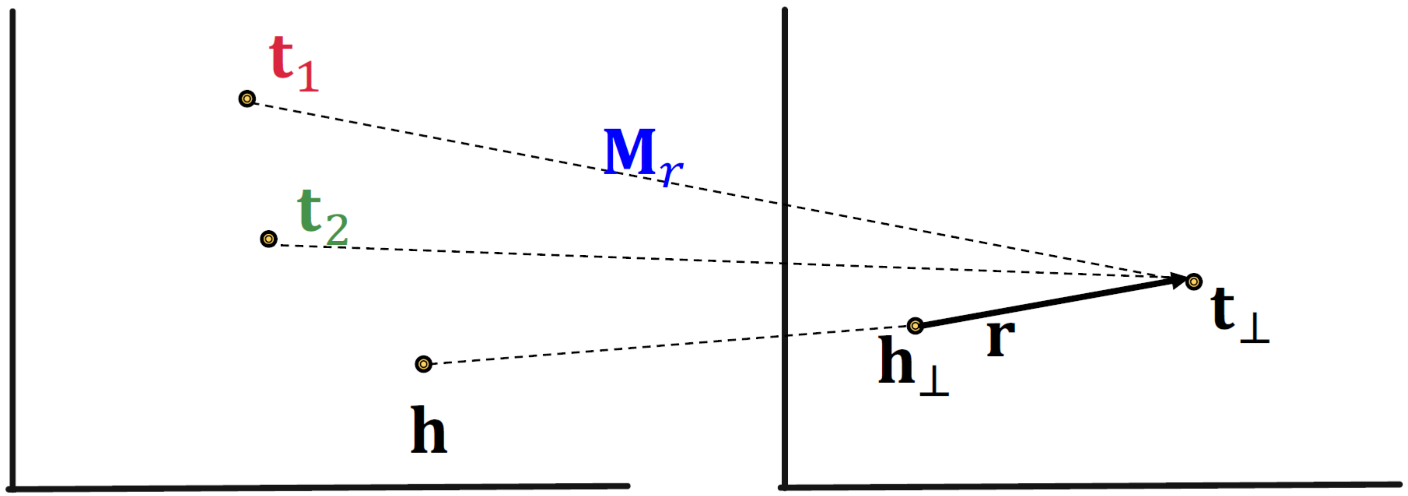
$$\mathbf{r} \neq 0, \mathbf{M}_r\mathbf{h} + \mathbf{r} = \mathbf{M}_r\mathbf{t},$$

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



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 $\mathbf{M}_r$ 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 $\mathbf{t}_2$!

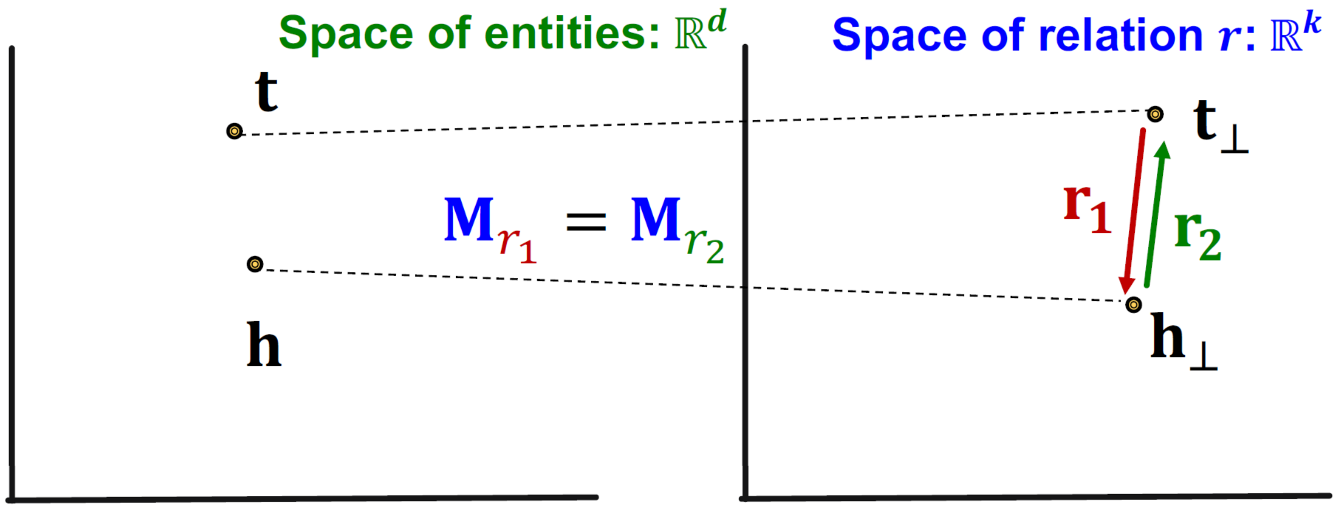


Inverse Relations in TransR

- **Inverse Relations:**
  - $r_1(h,t) \Rightarrow r_2(t,h)$
  - **Example** : (Advisor, Advisee)
- **TransR** can model inverse relations

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

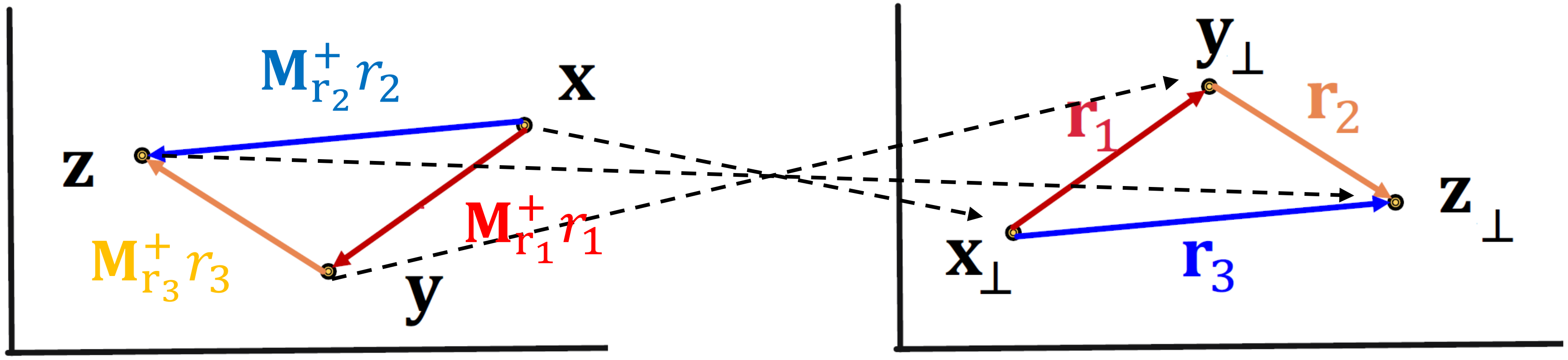
$$\mathbf{M}_{r_1}\mathbf{t} + \mathbf{r}_1 = \mathbf{M}_{r_1}\mathbf{h} \quad \mathbf{M}_{r_2}\mathbf{h} + \mathbf{r}_2 = \mathbf{M}_{r_2}\mathbf{t}$$



Composition Relations in TransR

- **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.
- **TransR** can model composition relations ✓

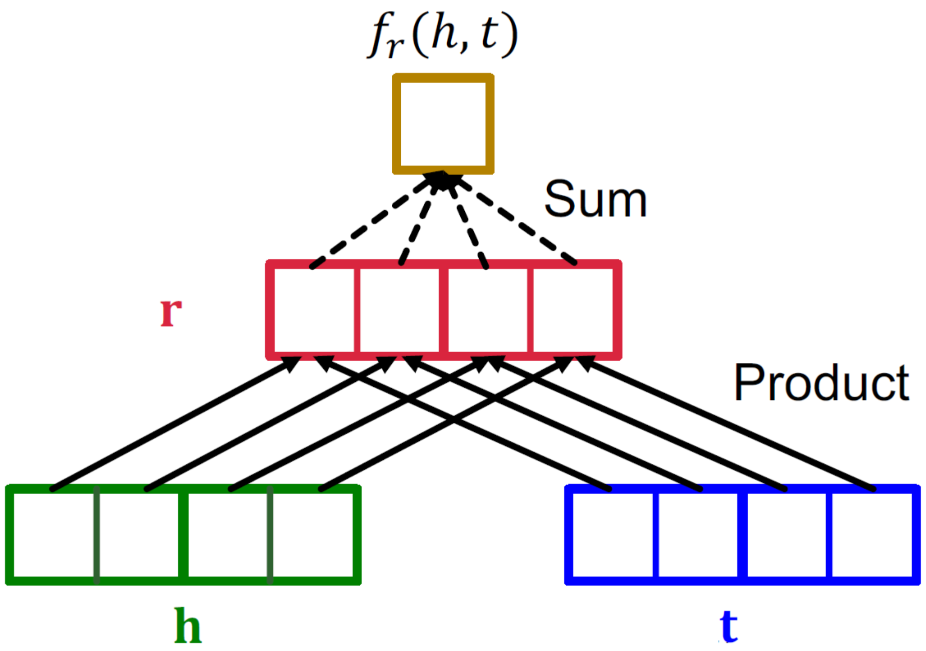
$\mathbf{M}_{\mathrm{r}_i}^{+} = \mathbf{M}_{\mathrm{r}_i}^{T}\left(\mathbf{M}_{\mathrm{r}_i}\mathbf{M}_{\mathrm{r}_i}^{T}\right)^{-1}$ is the Moore–Penrose inverse (pseudoinverse) of $\mathbf{M}_{\mathrm{r}_i}$ where $\mathbf{M}_{\mathrm{r}_i}\mathbf{M}_{\mathrm{r}_i}^{T}$ is invertible (full rank); so that $\mathbf{M}_{\mathrm{r}_i}\mathbf{M}_{\mathrm{r}_i}^{+} = \boldsymbol{I}$



https://inst.eecs.berkeley.edu/~ee127/sp21/livebook/def_pseudo_inv.html

DistMult

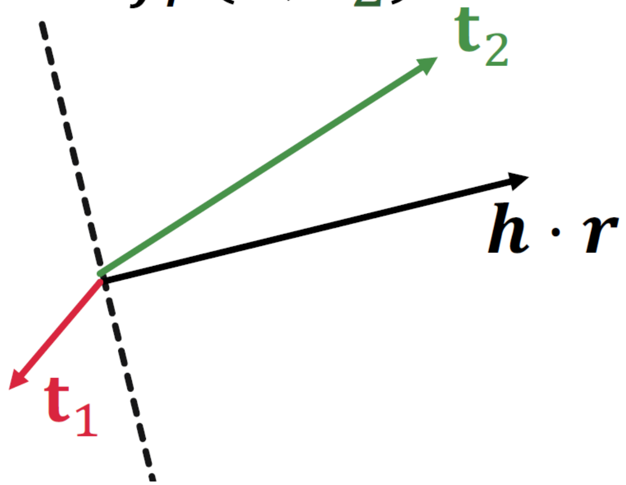
- **So far:** The scoring function $f_{\mathrm{r}}(h, t)$ is **negative of L1 / L2 distance** in **TransE** and **TransR**
- Another line of KG embeddings adopt **bilinear** modelling
- **DistMult**: Entities and relations using vectors in $\mathbb{R}^{\mathrm{k}}$
- **Score funcMon:** $f_{\mathrm{r}}(h,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



DistMult

- **Intuition of the score function:** Can be viewed as a **dot product** between $\mathbf{h} \cdot \mathbf{r}$ and $\mathbf{t}$
- **Example:**

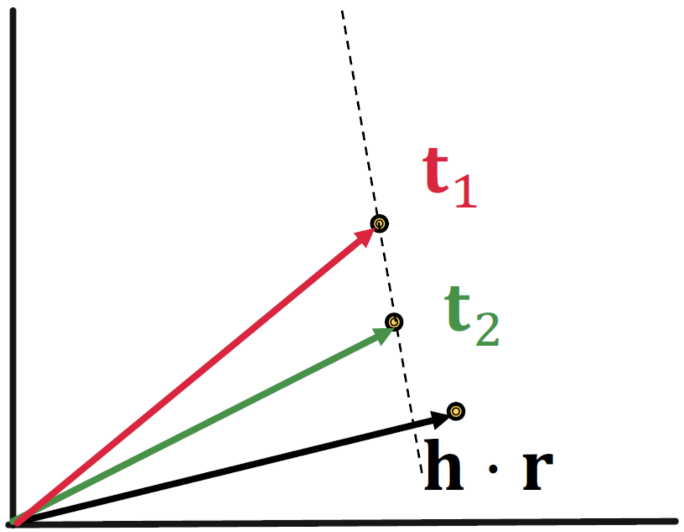
$$f_r(h, t_1) < 0, \qquad f_r(h, t_2) > 0$$



1-to-N Relations in DistMult

- **1-to-N Relations:**
  - **Example**: (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 ✓

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



Symmetric Relations in DistMult

- **Symmetric Relations:**
  - $r(h, t) \Rightarrow r(t, h) \forall h, t$
- **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: Inverse Relations

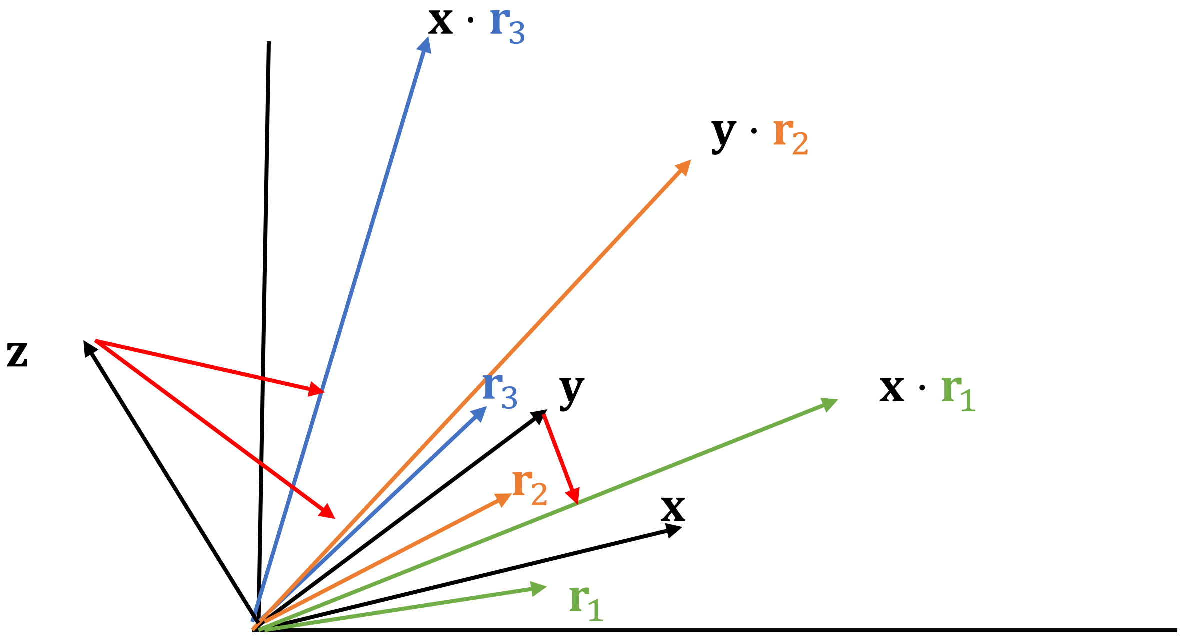
- **Inverse Relations:**
  - $r_1(h, t) \Rightarrow r_2(t, h)$
  - **Example** : (Advisor, Advisee)
- **DistMult** cannot model inverse relations
  - If it does model inverse relations:

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

  - This means $r_1 = r_2$
  - But semantically this does not make sense: **The embedding of "Advisor" should not be the same with "Advisee".**

Limitation: Composition Relations

- **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.
- **DistMult cannot** model composition relations
- **Intuition: DistMult** defines a hyperplane for each (head, relation), the union of the hyperplanes induced by multihops of relations, e.g., (r_1, r_2), cannot be expressed using a single hyperplane.



Some Other KG Embedding Algorithms

Model	Sym.	Antisym.	Inv.	Compos.	1-to-N
TransE	✗	✓	✓	✓	✗
TransR	✓	✓	✓	✓	✓
DistMult	✓	✗	✗	✗	✓
ComplEx	✓	✓	✓	✗	✓
RotatE	✓	✓	✓	✓	✓

https://arxiv.org/pdf/1902.10197.pdf

KG Embedding in Practice

- Different KGs may have drastically different relation patterns!
- There is not a general embedding that works for all KGs, use the **table** to select models
- Try **TransE** for a quick run if the target KG does not have much symmetric relations
- 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** models with different embedding space and expressiveness