

Knowledge Graph Embeddings

CPSC483: Deep Learning on Graph-Structured Data

Rex Ying

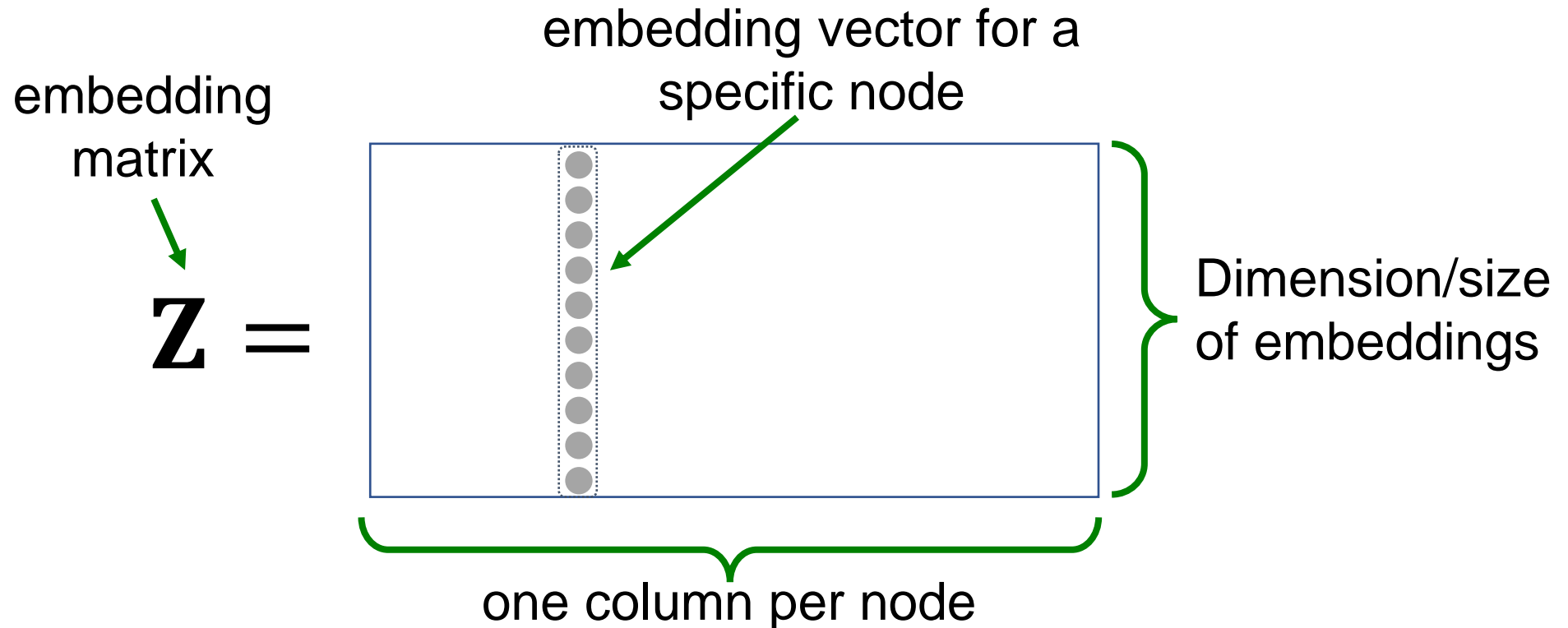
Readings

- Readings are updated on the website (syllabus page)
- **Lecture 17 readings:**
 - [Node2Vec](#)
 - [Graph Representation Learning survey](#)
- **Lecture 18 readings:**
 - [TransE: Translating Embeddings for Modeling Multi-relational Data](#)
 - [RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space](#)
 - [TuckER: Tensor Factorization for Knowledge Graph Completion](#)

Recap: Distributed Node Embeddings (1)

- **Shallow Encoding**


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



Recap: Distributed Node Embeddings (2)

An **unsupervised** setting for learning node embeddings

1. **Encoder ENC** maps from nodes to embeddings
2. Define a **node similarity function** (i.e., a measure of similarity in the original network)
3. **Decoder DEC** maps from embeddings to the similarity score
4. Optimize the parameters of the encoder so that:

$$\text{similarity}(u, v) \approx \mathbf{z}_v^T \mathbf{z}_u$$


Today: we define other **objectives** to learn node embeddings

Content

- **Introduction: Knowledge Graph**
- **Knowledge Graph Embedding Models**
 - TransE
 - DistMult & ComplEx
 - RotatE
 - TuckER

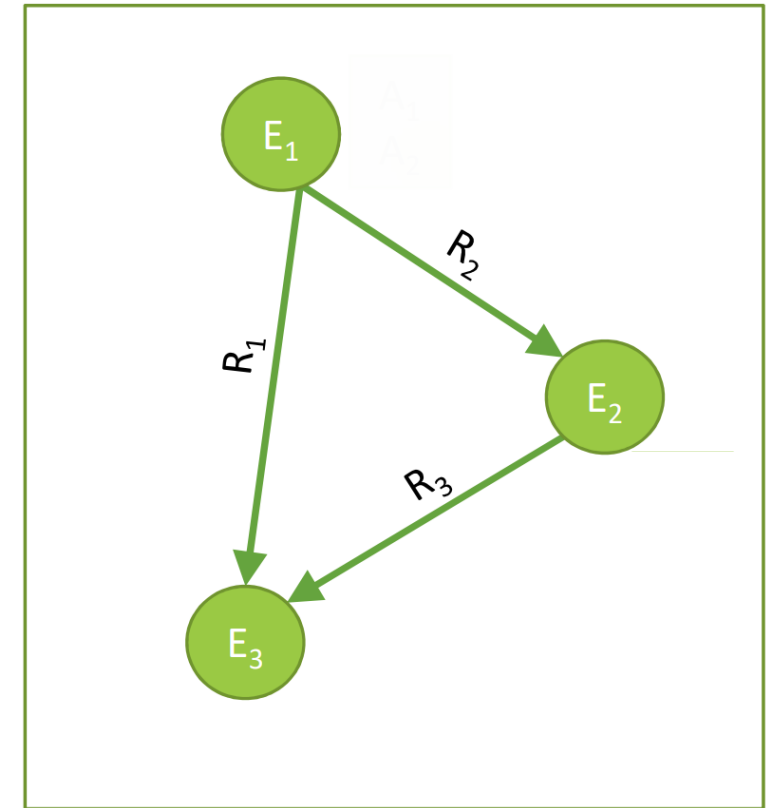
Content

- **Introduction: Knowledge Graph**
- **Knowledge Graph Embedding Models**
 - TransE
 - DistMult & ComplEx
 - RotatE
 - TuckER

Knowledge Graphs (KG)

Knowledge as a graph:

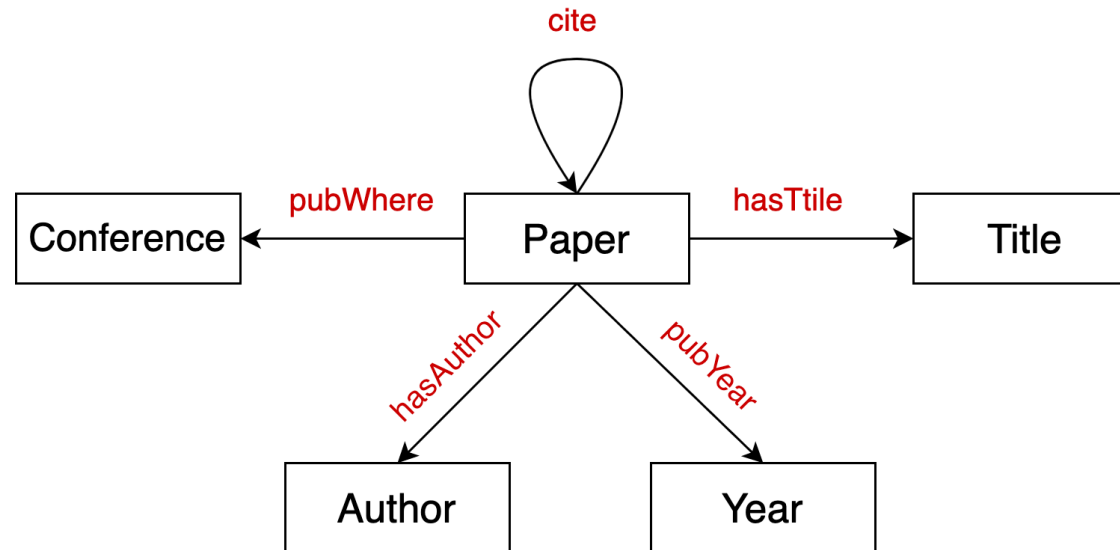
- A set of triplets <head entity, relationship, tail entity>
- Capture entities, types, and relationships
- Nodes are **entities**
- Nodes are labeled with their **types**
- Edges between two nodes capture **relationships** between entities
- **KG is an example of a heterogeneous graph**
 - **Recap:** Heterogeneous graph is a graph with multiple node types and edge types



E: entity
R: relation

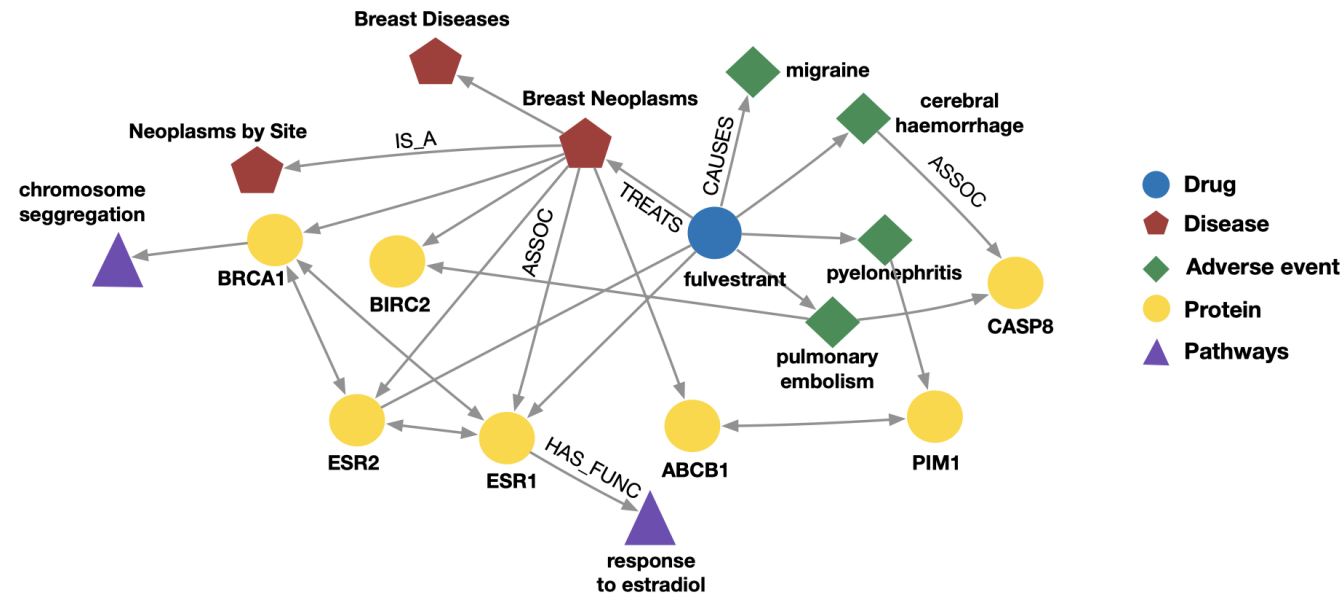
Example: Bibliographic Networks

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



Example: Biomedical 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 ([Knowledge Vault](#))
- Amazon Product Graph
- Facebook Graph API
- Project Hanover/Literome
- LinkedIn Knowledge Graph
- Yandex Object Answer
- NELL ([Never Ending Language Learning](#))

Applications of Knowledge Graphs

- Serving information

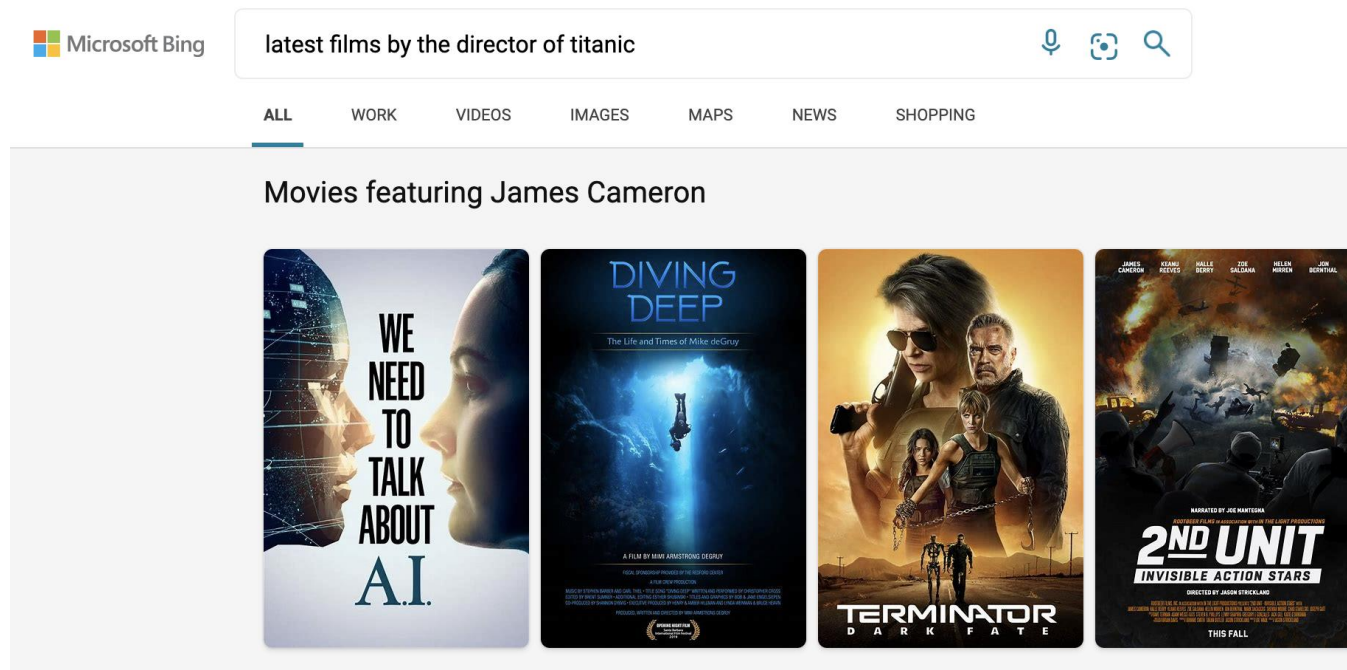


Image credit: Bing

Applications of Knowledge Graphs

- Question answering and conversation agents

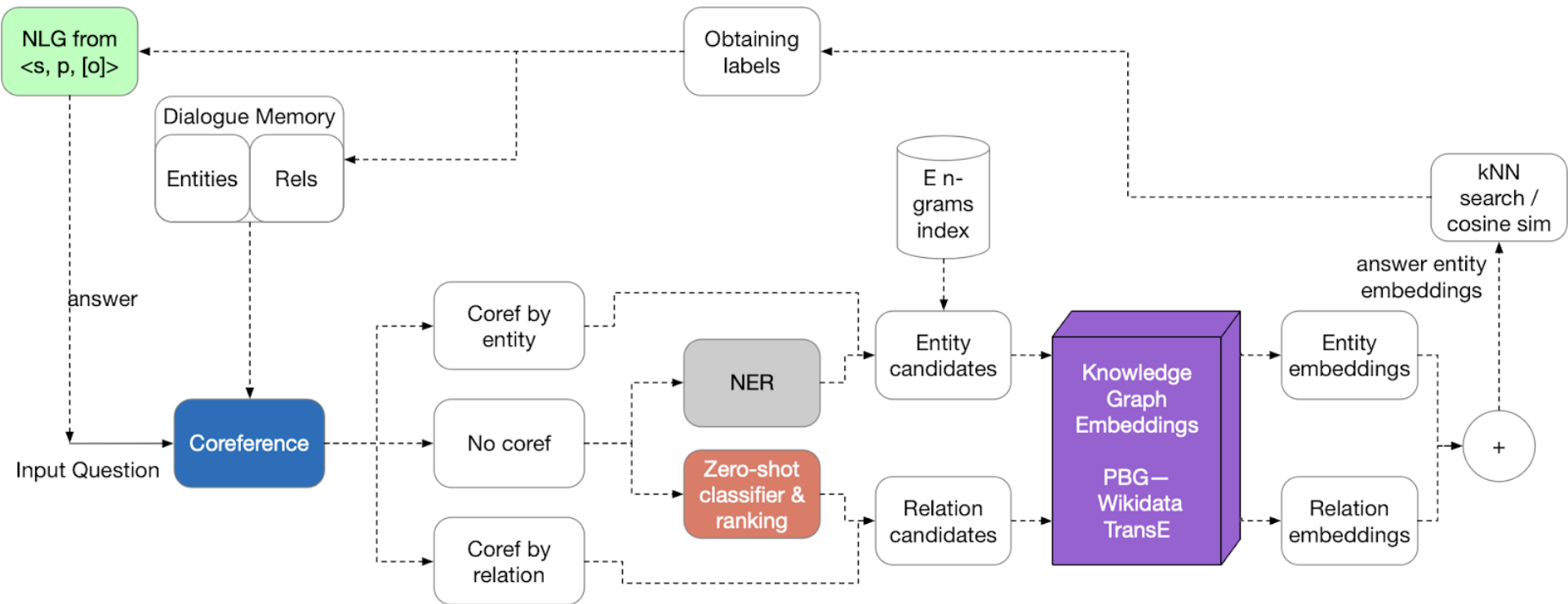
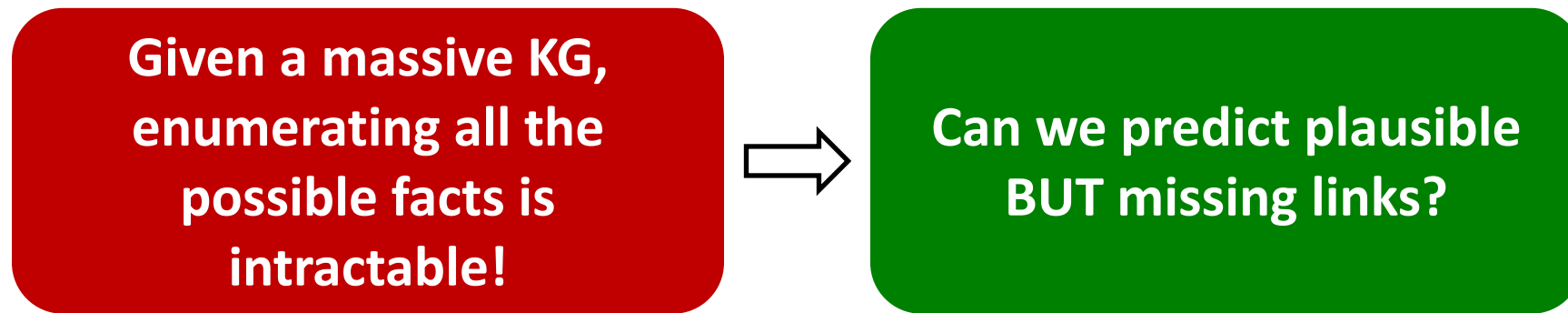


Image credit: [Medium](#)

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



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.

KG Completion Task

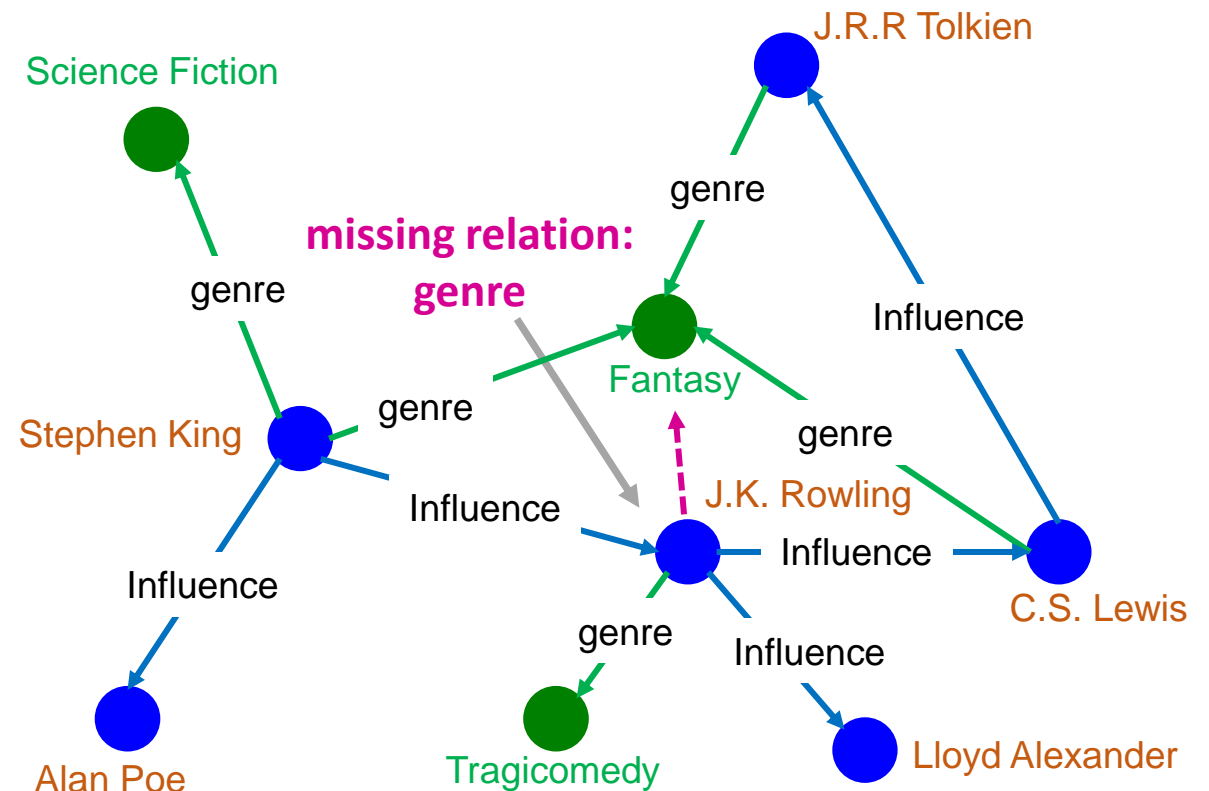
Knowledge graphs are usually **incomplete**. Many facts are missing!

Given an enormous KG, how can we complete the KG?

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

Recap: In link prediction task, we predict the edge $e = (u, v)$ based on the embeddings of head node u and tail node v

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



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**
 - **Deep encoder (GNN) is also possible here, using the same loss**
 - 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?

Content

- Introduction: Knowledge Graph
- **Knowledge Graph Embedding Models**
 - TransE
 - DistMult & ComplEx
 - RotatE
 - TuckER

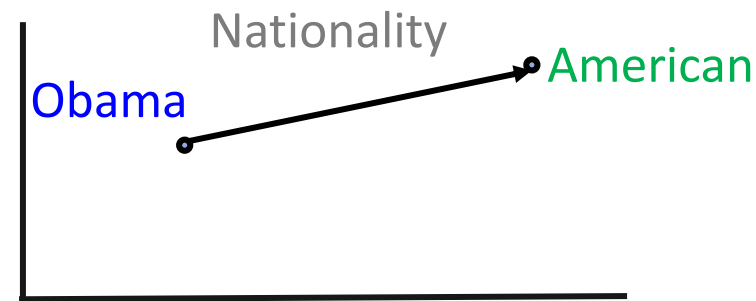
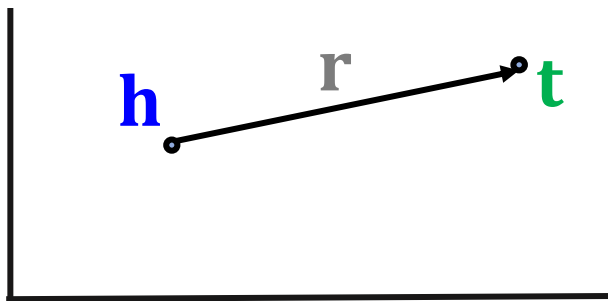
Translating Embeddings: TransE

- **Translation Intuition:**



For a triple (h, r, t) , $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$ are the embedding vectors
 $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ if the given fact is true
else $\mathbf{h} + \mathbf{r} \neq \mathbf{t}$

embedding vectors will
appear in boldface

- **Scoring function:** $f_r(h, t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||$
- Scoring function is **close to 0** if the fact is **true**!



Embedding Initialization

- Entities and relations are **initialized uniformly and then normalized**
- Given Training set $S = \{(h, r, t)\}$, where $h, t \in E, r \in R$.
(**entity set** E , **relation set** R , **embedding dimension** k)
- For each r in the **relation set** R ,
 - initialize $r \sim \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$  Xavier Initialization
(see more details in original [paper](#))
 - $r = r / \|r\|$  Normalization
- For each e in the **entity set** E ,
 - initialize $e \sim \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$
 - $e = e / \|e\|$

Negative Sampling

- **Negative sampling** with triplet that does not appear in the KG
- Set of **corrupted triplets**:

$$S'_{(h,l,t)} = \{(h', r, t) | h' \in E\} \cup \{(h, r, t') | t' \in E\}$$

replace **either the head or tail entity** by a random entity in **entity set E** (but not both at the same time) in training triplets

What will happen if no negative examples are provided?

TransE Training

- **Translation Intuition:** for a triple (h, r, t) ,
 $\mathbf{h} + \mathbf{r} = \mathbf{t}$

Max margin loss:

$$\mathcal{L} = \sum_{(h,r,t) \in G} \sum_{(h',r,t') \in S'_{(h,r,t)}} [\gamma - f_r(h, t) + f_r(h', t')]_+$$

Annotations:

- 0 if the term is negative (pointing to the $[\dots]_+$ term)
- Valid triple (pointing to (h, r, t))
- Corrupted triple (negative example) (pointing to (h', r, t'))
- corrupted set of (h, r, t) (pointing to $S'_{(h,r,t)}$)

where γ is the margin, i.e., **the smallest distance tolerated** by the model between a valid triple and a corrupted one.

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:**

Symmetric: $r(h, t) \Rightarrow r(t, h)$ (**Antisymmetric:** $r(h, t) \Rightarrow \neg r(t, h)$) $\forall h, t$

- **Example:**

- Symmetric: Family, Roommate

- **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:**

$$r(h, t_1), r(h, t_2), \dots, r(h, t_n) \text{ are all True.}$$

- **Example:** r is "StudentsOf"

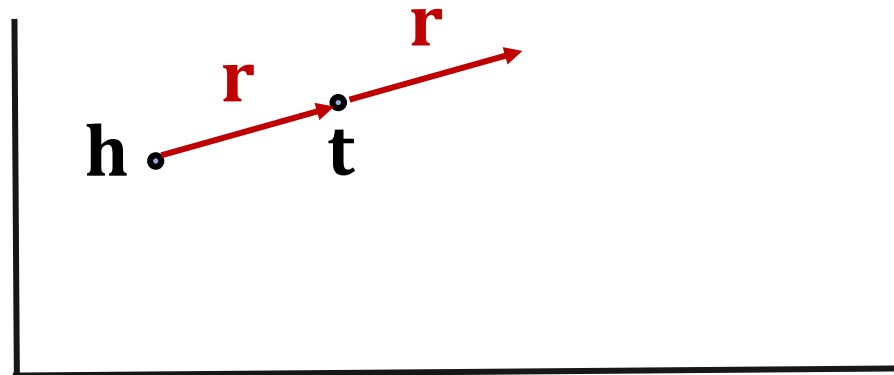
Antisymmetric Relations in TransE

- **Antisymmetric Relations:**

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

- **TransE** can model antisymmetric relations ✓

- $\mathbf{h} + \mathbf{r} = \mathbf{t}$, but $\mathbf{t} + \mathbf{r} \neq \mathbf{h}$



Inverse Relations in TransE

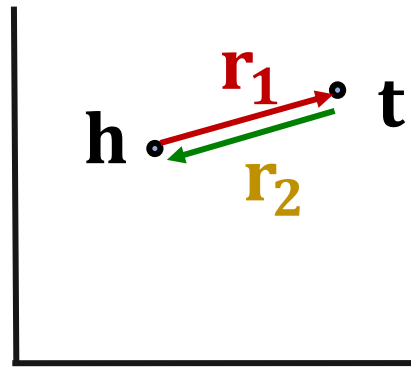
- **Inverse Relations:**

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

- **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

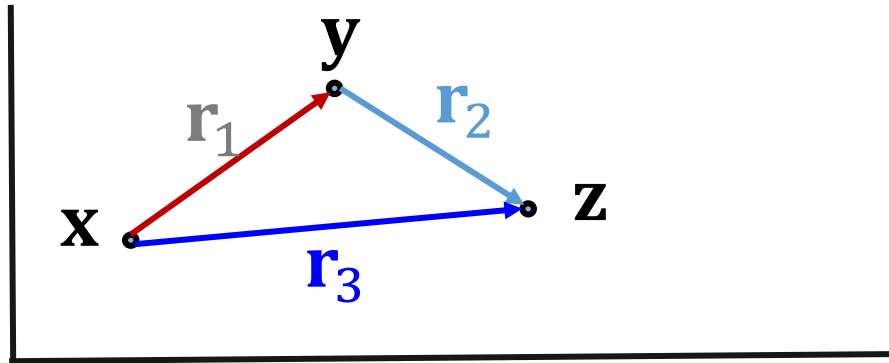
- **Composition Relations:**

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

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

- **TransE** can model composition relations ✓

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



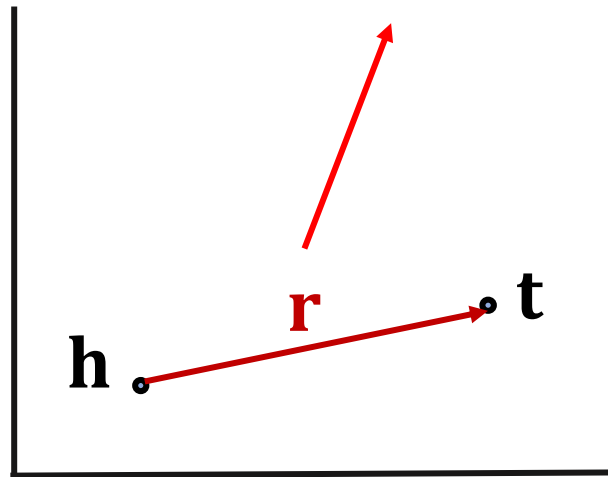
Limitation: Symmetric Relations

- **Symmetric Relations:**

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

- **Example:** Family, Roommate

- **TransE cannot** model symmetric relations **x**
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:**

- (h, r, t_1) and (h, r, t_2) both exist in the knowledge graph
- **Example:** r is “StudentsOf”

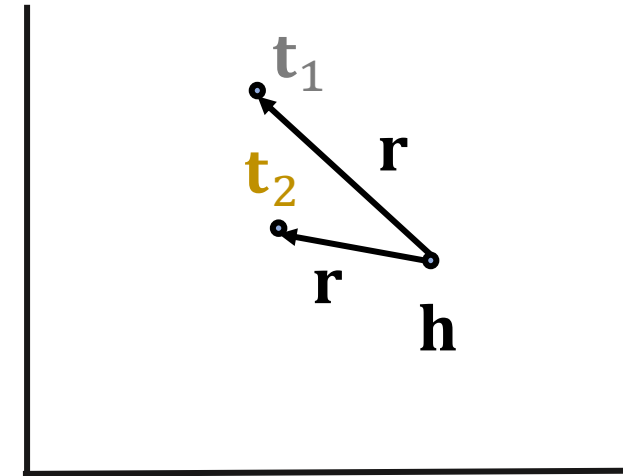
- **TransE cannot** model 1-to-N relations **✗**

- 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!

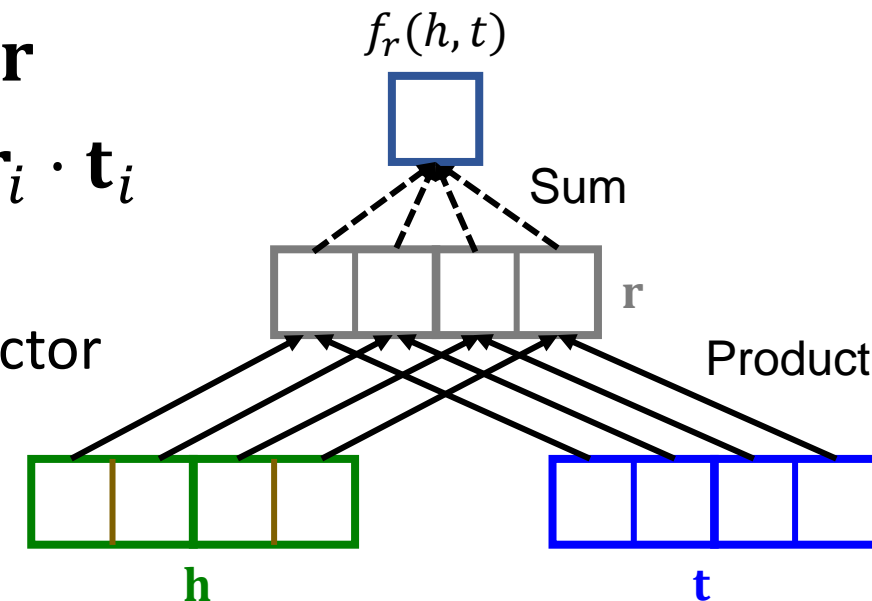


Content

- Introduction: Knowledge Graph
- **Knowledge Graph Embedding Models**
 - TransE
 - **DistMult & ComplEx**
 - RotatE
 - TuckER

New Idea: Bilinear Modeling

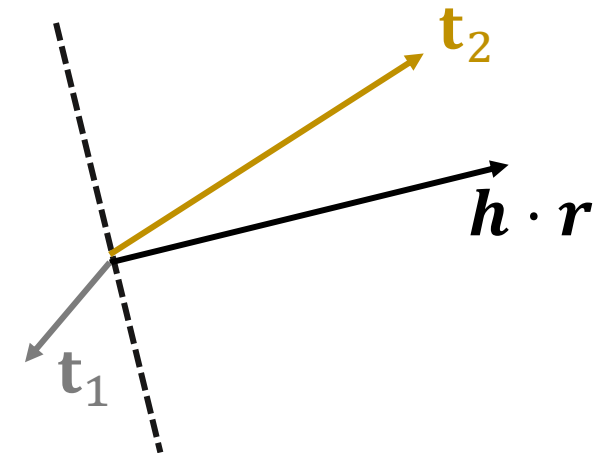
- **So far:** The scoring function $f_r(h, t)$ is the **negative L2 distance**
- **DistMult** considers **bilinear** modeling
- Entities are embedded as vectors in \mathbb{R}^k
- **Score function:** $f_r(h, t) = \mathbf{h}^T M_r \mathbf{t}$
 - M_r is a learnable matrix parameter for relation \mathbf{r}
- **A simpler version:** $f_r(h, t) = \langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle = \sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i$
 - Let M_r be a diagonal matrix
 - $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$ and i denotes the i -th dimension in the vector



DistMult

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

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



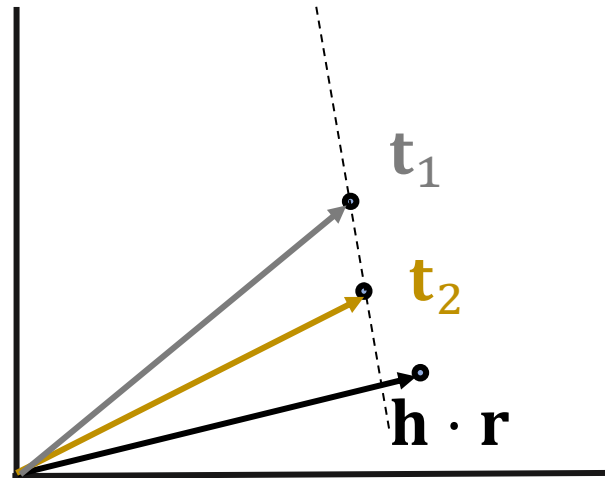
1-to-N Relations in DistMult

- **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 ✓

$$\langle \mathbf{h}, \mathbf{r}, \mathbf{t}_1 \rangle = \langle \mathbf{h}, \mathbf{r}, \mathbf{t}_2 \rangle$$



The projection of \mathbf{t}_1 and \mathbf{t}_2 along the $\mathbf{h} \cdot \mathbf{r}$ direction is the same

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

$$f_r(h, t) = \langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle = \sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i = \\ \langle \mathbf{t}, \mathbf{r}, \mathbf{h} \rangle = f_r(t, h) \quad \checkmark$$

Limitation: Antisymmetric Relations

- **Antisymmetric Relations:**

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

- **Example:** Hypernym

- **DistMult cannot** model antisymmetric relations

$$f_r(h, t) = \langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle = \langle \mathbf{t}, \mathbf{r}, \mathbf{h} \rangle = f_r(t, h) \quad \times$$

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

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:

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

- 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

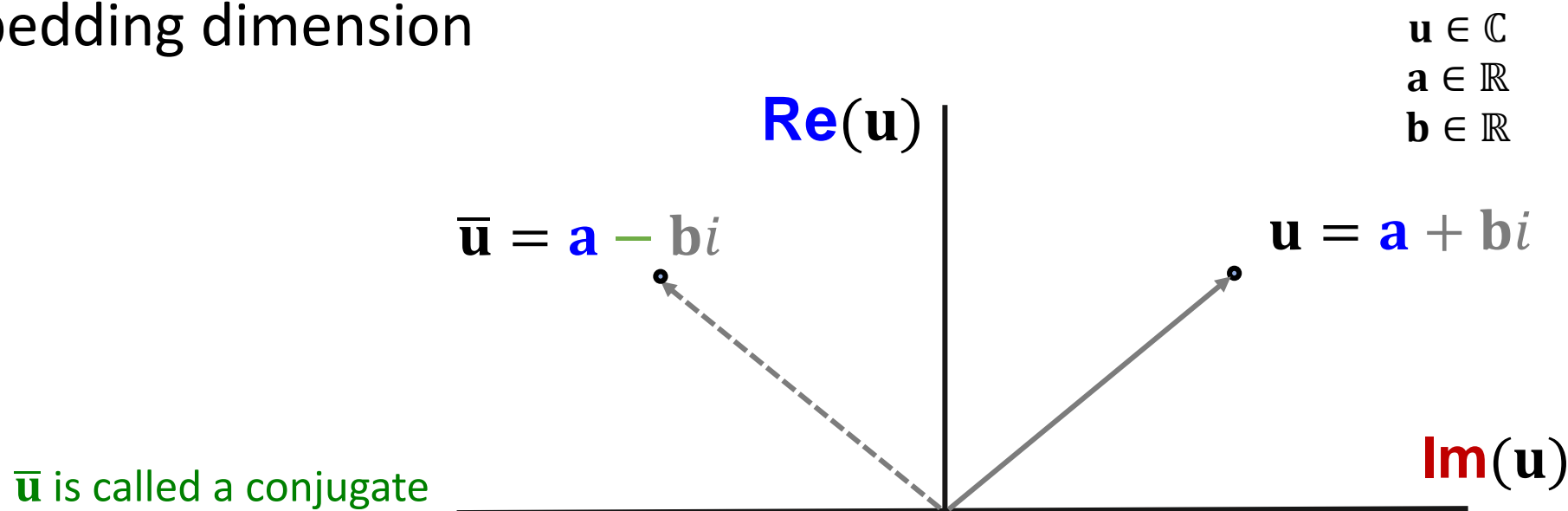
- **Composition Relations:**

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

- **Example:** My mother's husband is my father.
- **DistMult** can model composition relations ✓
 - DistMult uses matrix multiplication / inner product
 - Composition can correspond to chain matrix multiplication

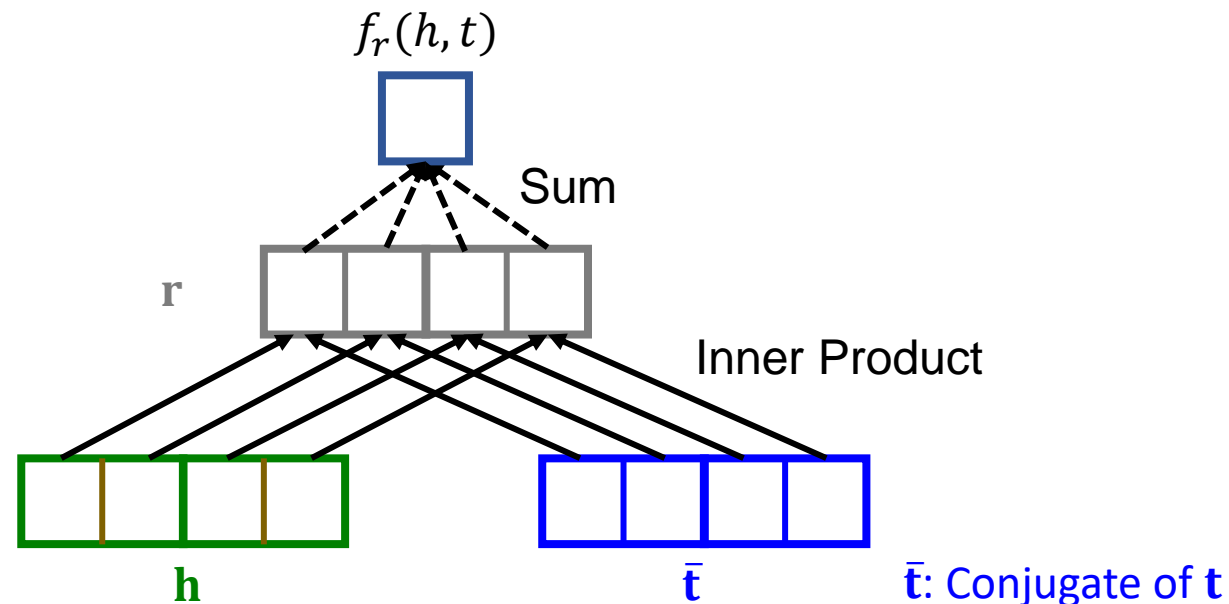
ComplEx

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



ComplEx

- **ComplEx**: model entities and relations using vectors in \mathbb{C}^k
- For each triple (h, r, t) , $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^k$ are the embedding vectors
- **Score function** $f_r(h, t) = \text{Re}(\sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{t}}_i)$



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 $\text{Im}(\mathbf{r}) = \mathbf{0}$, we have**

- $$\begin{aligned} f_r(h, t) &= \text{Re}(\sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{t}}_i) = \sum_i \text{Re}(\mathbf{r}_i \cdot \mathbf{h}_i \cdot \bar{\mathbf{t}}_i) \\ &= \sum_i \mathbf{r}_i \cdot \text{Re}(\mathbf{h}_i \cdot \bar{\mathbf{t}}_i) = \sum_i \mathbf{r}_i \cdot \text{Re}(\bar{\mathbf{h}}_i \cdot \mathbf{t}_i) \\ &= \sum_i \text{Re}(\mathbf{r}_i \cdot \bar{\mathbf{h}}_i \cdot \mathbf{t}_i) = f_r(t, h) \end{aligned}$$

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
 - $f_r(h, t) = \text{Re}(\sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{t}}_i)$
 - $f_r(t, r) = \text{Re}(\sum_i \mathbf{t}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{h}}_i)$
 - $r(h, t)$ and $r(t, h)$ can have different scores in ComplEx, due to the asymmetric modeling using complex conjugate.

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 $r_2 = \operatorname{argmax}_{\mathbf{r}} \operatorname{Re}(\langle \mathbf{h}, \mathbf{r}, \bar{\mathbf{t}} \rangle)$:

\bar{r}_2 is exactly $r_1 = \operatorname{argmax}_{\mathbf{r}} \operatorname{Re}(\langle \mathbf{t}, \mathbf{r}, \bar{\mathbf{h}} \rangle)$.

- Note: $\operatorname{Re}(\langle \mathbf{h}, \mathbf{r}, \bar{\mathbf{t}} \rangle) = \operatorname{Re}(\langle \mathbf{t}, \bar{\mathbf{r}}, \bar{\mathbf{h}} \rangle) \Rightarrow r_1 = \bar{r}_2$

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** shares the same property with **DistMult**

Content

- Introduction: Knowledge Graph
- **Knowledge Graph Embedding Models**
 - TransE
 - DistMult & ComplEx
 - **RotatE**
 - TuckER

RotatE

- **RotatE**: represent triple (h, r, t) in **complex vector space**, i.e., $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^k$
- k is the embedding dimension
- Define each relation \mathbf{r} as an element-wise rotation from the head \mathbf{h} to the tail \mathbf{t}

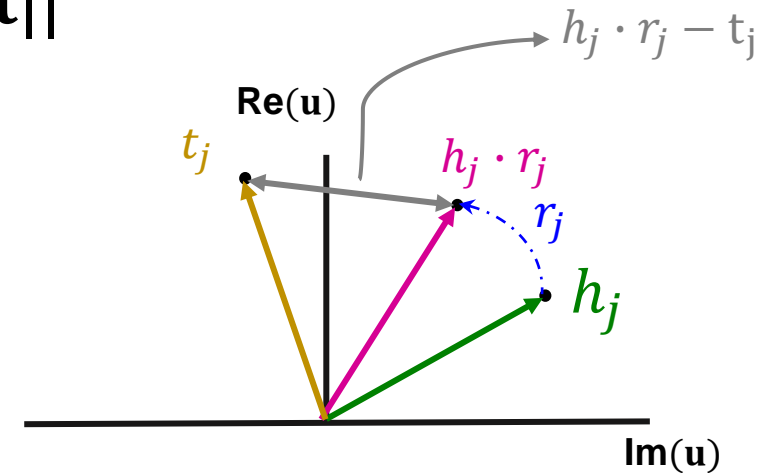
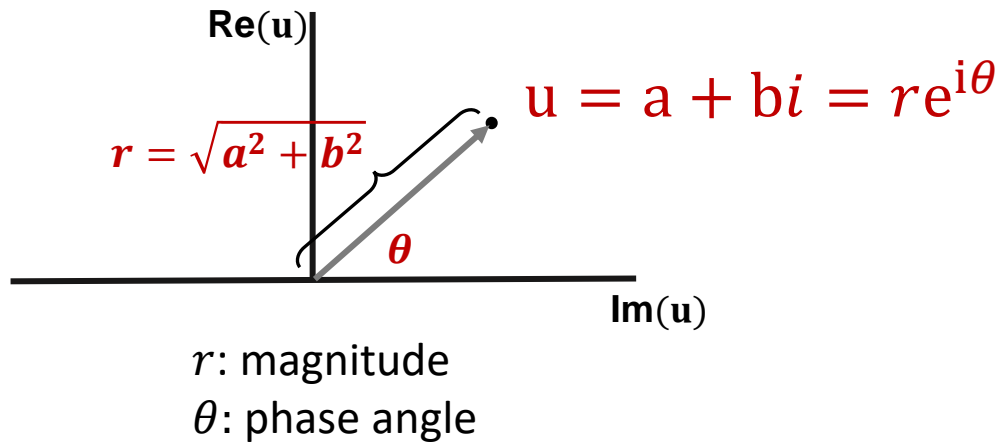
$$\mathbf{t} = \mathbf{h} \circ \mathbf{r}, \text{ where } |r_j| = 1$$

- \circ is the **Hadamard** (element-wise) product (i.e., $t_j = h_j \cdot r_j$ for each j).
- j denotes the j -th dimension, $t_j, h_j, r_j \in \mathbb{C}$.

Relation as Rotation

- Since $|r_j| = 1$, r_j can also be represented as $r_j = e^{i\theta_{r,j}}$, $\theta_{r,j}$ is the **phase angle**
- $r_j = e^{i\theta_{r,j}}$ and $t_j = h_j \cdot r_j$ means h_j is **rotated $\theta_{r,j}$ clockwise** in the complex space
- Define **the score function** of RotatE as

$$f_r(\mathbf{h}, \mathbf{t}) = -||\mathbf{h} \circ \mathbf{r} - \mathbf{t}||$$



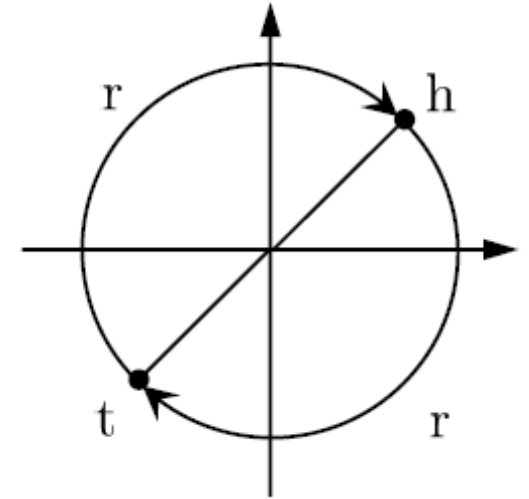
Symmetric and Antisymmetric in RotatE

- RotatE can model symmetric and antisymmetric Relations

- **Symmetric** Relations:

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

- r is **symmetric** if and only if $r_j = \pm 1$, i.e., $\theta_{r,j} = 0$ or π
- An example on the space of \mathbb{C}
 - $r_j = -1$ or $\theta_{r,j} = \pi$



- **Antisymmetric** Relations: $r(h, t) \Rightarrow \neg r(t, h) \quad \forall h, t$
- r is **antisymmetric** if and only if $\mathbf{r} \circ \mathbf{r} \neq \mathbf{1}$, i.e., $\theta_{r,j} \neq 0$ and π

Other Relations in RotatE

- RotatE can model Inverse and Composition relations
- **Inverse Relations:** $r_2(h, t) \Rightarrow r_1(t, h)$
- Two relations r_1 and r_2 are **inverse** if and only if $\mathbf{r}_2 = \bar{\mathbf{r}}_1$, i.e., $\theta_{2,j} = -\theta_{1,j}$
- **Composition Relations:** $r_1(x, y) \wedge r_2(y, z) \Rightarrow r_3(x, z) \quad \forall x, y, z$
- relation $\mathbf{r}_3 = \mathbf{e}^{i\theta_3}$ is a **composition** of two relations $\mathbf{r}_1 = \mathbf{e}^{i\theta_1}$ and $\mathbf{r}_2 = \mathbf{e}^{i\theta_2}$ if only if $\mathbf{r}_3 = \mathbf{r}_1 \circ \mathbf{r}_2$,
 - i.e., $\theta_{3,j} = \theta_{1,j} + \theta_{2,j}$.
 - $\theta_{k,j}$ is the k -th dimension of $\boldsymbol{\theta}_k$

Limitations of RotatE

- **1-to-N Relations:**

- **Example:** If (h, r, t_1) and (h, r, t_2) exist in the knowledge graph

- RotatE cannot model 1-to-N relations

- if $\mathbf{t}_1 = \mathbf{h} \circ \mathbf{r}$ and $\mathbf{t}_2 = \mathbf{h} \circ \mathbf{r}$,

$$\Rightarrow t_{1,j} = h_j \cdot r_j; t_{2,j} = h_j \cdot r_j \text{ for each } j$$

$$\Rightarrow \mathbf{t}_1 = \mathbf{t}_2!$$

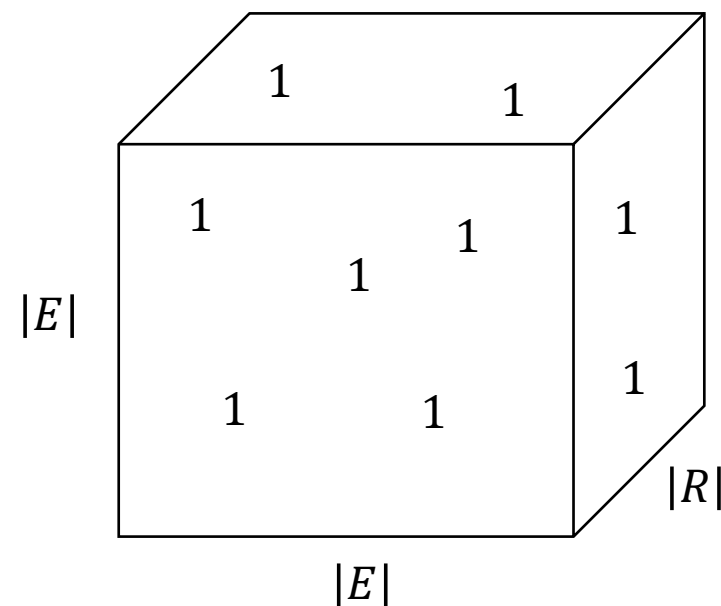
Content

- Introduction: Knowledge Graph
- **Knowledge Graph Embedding Models**
 - TransE
 - DistMult & ComplEx
 - RotatE
 - **TuckER**

TuckER

- An alternative view of a knowledge graph: **sparse binary adjacency tensor** representation of known fact.
- E is the entity set, R is the relation set
- If (h, r, t) exists in KG, then

$$\text{tensor}(h, r, t) = 1$$



Background: Tucker Decomposition

- **Tucker decomposition** decomposes a tensor into a set of matrices and **a smaller core tensor**.
- In three-mode case, given the original tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, Tucker decomposition outputs a tensor $\mathcal{Z} \in \mathbb{R}^{P \times Q \times R}$ and three matrices $\mathbf{A} \in \mathbb{R}^{I \times P}$, $\mathbf{B} \in \mathbb{R}^{J \times Q}$, $\mathbf{C} \in \mathbb{R}^{K \times R}$:

$$\mathcal{X} \approx \mathcal{Z} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$$

- \times_n indicates the tensor product along the n -th mode
- Elements of the **core tensor** \mathcal{Z} , $\mathcal{Z}(p, q, r)$ show **the interaction level** between the component $\mathbf{A}[p]$, $\mathbf{B}[q]$ and $\mathbf{C}[r]$. $[i]$ denotes the i -th column
- the **core tensor** \mathcal{Z} can be considered as **a compressed version of \mathcal{X}**

Tucker decomposition for KG

- **TuckER** uses **Tucker decomposition** for Knowledge Graph Completion
- entity embedding matrix $\mathbf{E} \in \mathbb{R}^{n_e \times d_e}$ and relation embedding matrix $\mathbf{R} \in \mathbb{R}^{n_r \times d_r}$
 - n_e, n_r represent the number of entities and relations
 - d_e, d_r represent the dimensionality of entity embedding and relation embedding
- Three matrices in Tucker: $\mathbf{A} = \mathbf{C} = \mathbf{E} \in \mathbb{R}^{n_e \times d_e}$ and $\mathbf{B} = \mathbf{R} \in \mathbb{R}^{n_r \times d_r}$
- **Scoring function for TuckER:**

$$\phi(\mathbf{h}, \mathbf{r}, \mathbf{t}) = \mathcal{W} \times_1 \mathbf{h} \times_2 \mathbf{r} \times_3 \mathbf{t}$$

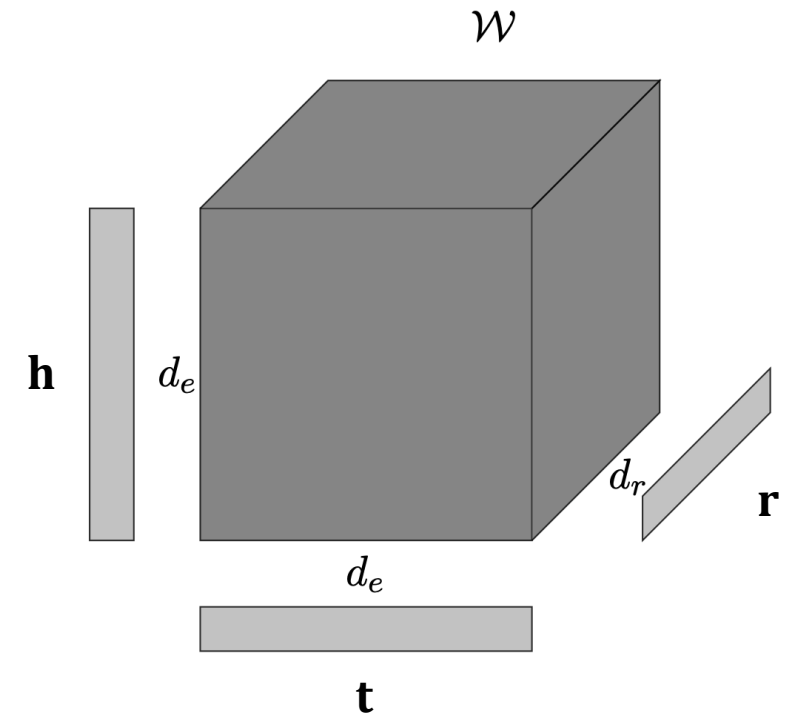
- $\mathbf{h}, \mathbf{t} \in \mathbb{R}^{d_e}$ are the rows of \mathbf{E} , $\mathbf{r} \in \mathbb{R}^{d_r}$ is the row of \mathbf{R}
- $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ is the core tensor
- d_e, d_r can be different in TuckER

TuckER architecture

- **Scoring function:**

$$\phi(\mathbf{h}, \mathbf{r}, \mathbf{t}) = \mathcal{W} \times_1 \mathbf{h} \times_2 \mathbf{r} \times_3 \mathbf{t}$$

- Apply **logistic sigmoid** to score $\phi(\mathbf{h}, \mathbf{r}, \mathbf{t})$ to obtain the predicted probability of triple being true
- The core tensor \mathcal{W} can be viewed as a **shared pool of “prototype” relation matrices**, which are linearly combined using the parameters in each relation embedding \mathbf{r} .



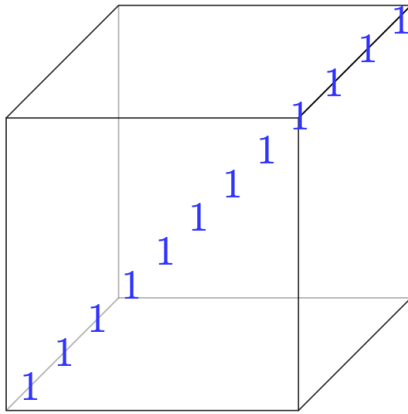
Relation to Previous Models

- Recap: Score function of **DistMulti**:

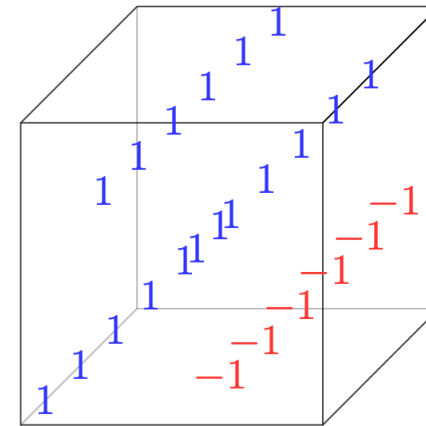
$$f_r(h, t) = \langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle = \sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i; \quad \mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$$

- Score function of **Complex**:

$$f_r(h, t) = \text{Re}(\sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{t}}_i); \quad \mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^k$$



Core tensor $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ for DistMult



Core tensor $\mathcal{W} \in \mathbb{R}^{2d_e \times 2d_r \times 2d_e}$ for Complex

Expressiveness of All Models

- Properties and expressive power of different KG completion methods:

Model	Score	Embedding	Sym .	Antisym.	Inv.	Compos.	1-to-N
TransE	$-\ \mathbf{h} + \mathbf{r} - \mathbf{t}\ $	$\mathbf{h}, \mathbf{t}, \mathbf{r} \in \mathbb{R}^k$	✗	✓	✓	✓	✗
DistMult	$\langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle$	$\mathbf{h}, \mathbf{t}, \mathbf{r} \in \mathbb{R}^k$	✓	✗	✗	✓	✓
Complex	$\text{Re}(\langle \mathbf{h}, \mathbf{r}, \bar{\mathbf{t}} \rangle)$	$\mathbf{h}, \mathbf{t}, \mathbf{r} \in \mathbb{C}^k$	✓	✓	✓	✓	✓
RotatE	$-\ \mathbf{h} \circ \mathbf{r} - \mathbf{t}\ $	$\mathbf{h}, \mathbf{t}, \mathbf{r} \in \mathbb{C}^k$	✓	✓	✓	✓	✗
Tucker	$\mathcal{W} \times_1 \mathbf{h} \times_2 \mathbf{r} \times_3 \mathbf{t}$	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^{d_e},$ $\mathbf{r} \in \mathbb{R}^{d_r}$	✓	✓	✓	✓	✓

- Complex extends DistMult by introducing complex embeddings
- RotatE can degenerate into TransE (See proof in [original paper, Appendix F](#))
- Tucker uses tensor factorization to propose a fully expressive knowledge graph model

Today's Summary

- Different KGs may have **drastically different relation patterns!**
- **Modeling relation patterns** is critical for knowledge base completion
 - Symmetric/Antisymmetric, Inverse, composition, 1-to-N
- There is not a general embedding that works for all KGs, use the **table** to select models
- In practice: try **TransE** for a quick run if the target KG does not have much symmetric relations, then use more expressive models, e.g., **DistMult**, **Complex**, **RotatE**, **TuckER**