Knowledge Graphs

ACMS 80770: Deep Learning with Graphs

Instructor: Navid Shervani-Tabar

Department of Applied and Comp Math and Stats

UNIVERSITY OF NOTRE DAME

Knowledge Graph

- Real world **information** can be stored in the form of **entities** and the **relationships** between them.
- To that end, a fact can be represented using triple

$$(subject, predicate, object)$$

  where $subject$ and $object$ are entities and $predicate$ is their relation.
- The **set** of these triples represents **knowledge**.

Knowledge Graph

- Consider the statement

  Matthew Perry is an actor who portrayed the character Chandler Bing in the television sitcom Friends.

Knowledge Graph

- Consider the statement

Matthew Perry is an actor who portrayed the character Chandler Bing in the television sitcom Friends.

- This information can be represented using the following set of triples.

Subject	Predicate	Object

Knowledge Graph

- Consider the statement

  Matthew Perry is an actor who portrayed the character Chandler Bing in the television sitcom Friends.

- This information can be represented using the following set of triples.

Subject	Predicate	Object
Matthew Perry	profession	Actor

Knowledge Graph

- Consider the statement

  Matthew Perry is an actor who portrayed the character Chandler Bing in the television sitcom Friends.

- This information can be represented using the following set of triples.

Subject	Predicate	Object
Matthew Perry	profession	Actor
Matthew Perry	starredIn	Friends

Knowledge Graph

- Consider the statement

Matthew Perry is an actor who portrayed the character Chandler Bing in the television sitcom Friends.

- This information can be represented using the following set of triples.

Subject	Predicate	Object
Matthew Perry	profession	Actor
Matthew Perry	starredIn	Friends
Matthew Perry	portrayed	Chandler Bing

Knowledge Graph

- Consider the statement

  Matthew Perry is an actor who portrayed the character Chandler Bing in the television sitcom Friends.

- This information can be represented using the following set of triples.

Subject	Predicate	Object
Matthew Perry	profession	Actor
Matthew Perry	starredIn	Friends
Matthew Perry	portrayed	Chandler Bing
Chandler Bing	characterIn	Friends

Knowledge Graph

❖ Consider the statement

Matthew Perry is an actor who portrayed the character Chandler Bing in the television sitcom Friends.

❖ This information can be represented using the following set of triples.

Subject	Predicate	Object
Matthew Perry	profession	Actor
Matthew Perry	starredIn	Friends
Matthew Perry	portrayed	Chandler Bing
Chandler Bing	characterIn	Friends
Friends	genre	Sitcom

Knowledge Graph

- Alternatively, one can **combine** these triples and represent them using a **graph structure**.
- In this representation, we show **entities** of interest with nodes and the **relations** between these entities using edges.

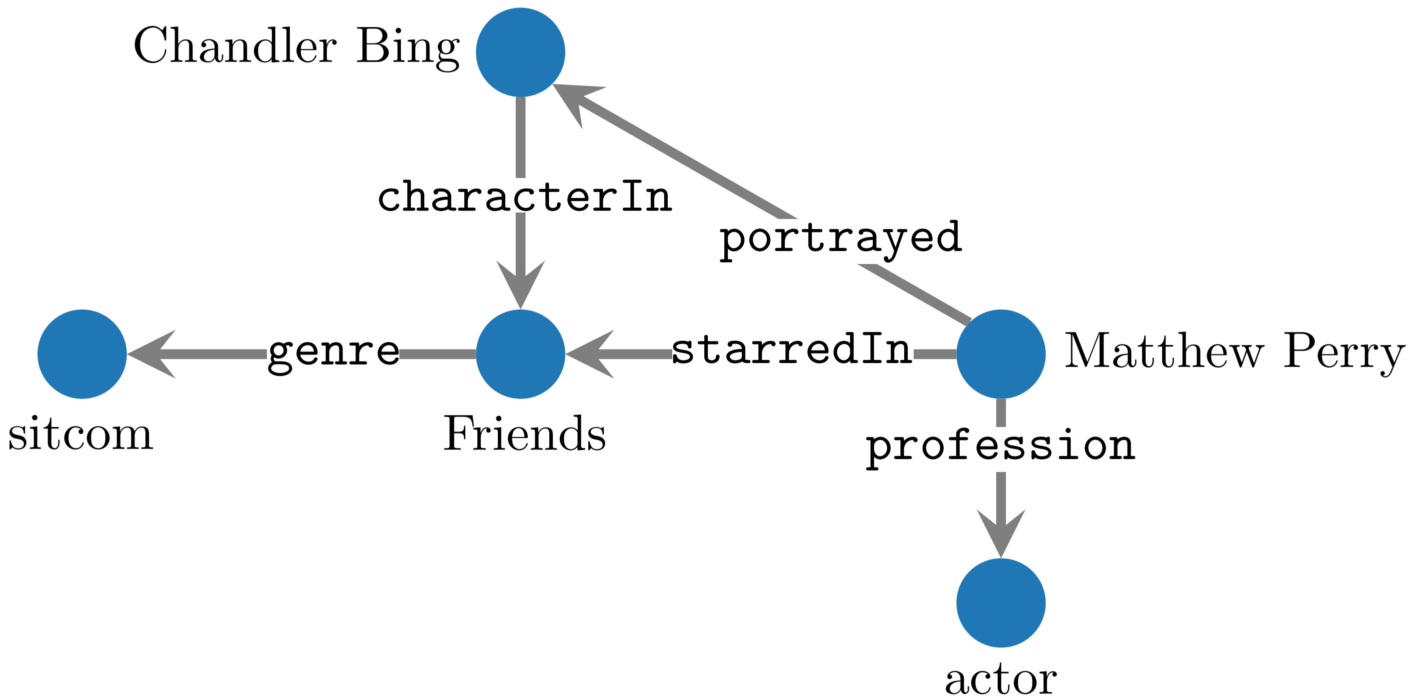
Knowledge Graph

- Alternatively, one can **combine** these triples and represent them using a **graph structure**.
- In this representation, we show **entities** of interest with nodes and the **relations** between these entities using edges.

- Each 3-tuple (v_h, τ, v_t) specifies a **fact** that holds between nodes v_h and v_t.
- This graph-structured representation of information is known as **knowledge graph**.
- Knowledge graphs can be used to represent the **factual information** and the **knowledge of the real world**.

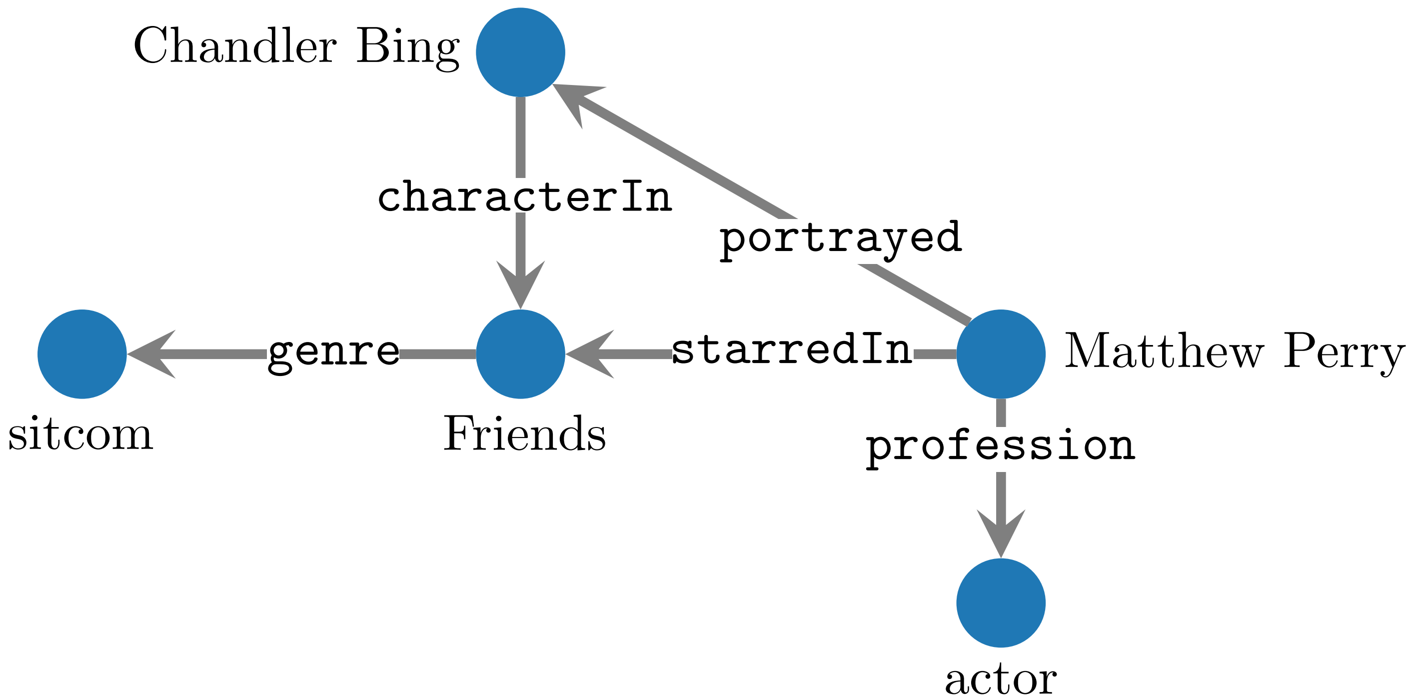
Knowledge Graph

➢ We can represent the set of triplets in our example in a **graph-structured** manner.



Knowledge Graph

- We can represent the set of triplets in our example in a **graph-structured** manner.



- **Knowledge graphs** are consist of various types of nodes and edges and therefore are **heterogenous** graphs.

Heterogeneous Graphs

- Heterogeneous graphs are constructed of **nodes** and **edges** that have **different types**.
- Each edge is defined by $(u, \tau, v) \in \mathrm{E}$, where τ is the type of the edge.

Heterogeneous Graphs

- Heterogeneous graphs are constructed of **nodes** and **edges** that have **different types**.
- Each edge is defined by $(u, \tau, v) \in \mathrm{E}$, where τ is the type of the edge.
- In a heterogeneous graph

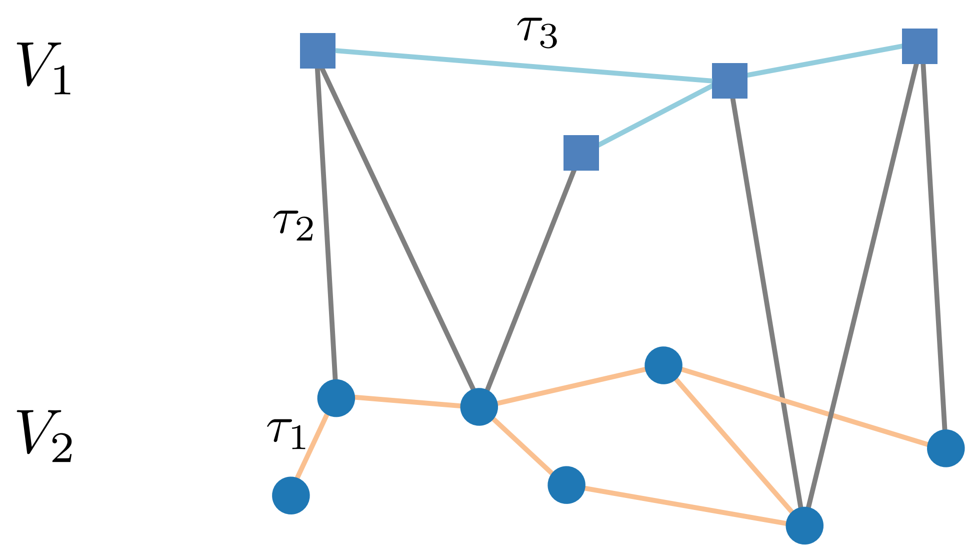
$$V = V_1 \cup \ldots \cup V_k, V_i \cap V_j = \emptyset, \forall i \neq j$$

  Here, V_j is set of **nodes** with **type** j.
- Edges of specific type τ_i usually only connect nodes of certain types.

$$(u, \tau_i, v) \in \varepsilon \rightarrow u \in V_j, v \in V_k$$

Heterogeneous Graphs

- Polypharmacy side-effect graph



τ_1 : Protein-protein interaction

τ_2 : Protein-drug interaction

τ_3 : Polypharmacy side-effect

V_1 : Drugs

V_2 : Proteins

Knowledge Graph

- In recent years, a large number of Knowledge graphs have been created.

Knowledge Graph

- In recent years, a large number of Knowledge graphs have been created.

- DBpedia: Wikipedia

Knowledge Graph

- In recent years, a large number of Knowledge graphs have been created.
- DBpedia: Wikipedia
- Freebase: Human contributor

Knowledge Graph

- In recent years, a large number of Knowledge graphs have been created.

- DBpedia: Wikipedia
- Freebase: Human contributor
- Commerce: Amazon, Uber, Airbnb

Knowledge Graph

- In recent years, a large number of Knowledge graphs have been created.

- DBpedia: Wikipedia
- Freebase: Human contributor
- Commerce: Amazon, Uber, Airbnb
- Finance: Bloomberg, Capital One.

Knowledge Graph

- In recent years, a large number of Knowledge graphs have been created.
- DBpedia: Wikipedia
- Freebase: Human contributor
- Commerce: Amazon, Uber, Airbnb
- Finance: Bloomberg, Capital One.
- Social Media: LinkedIn

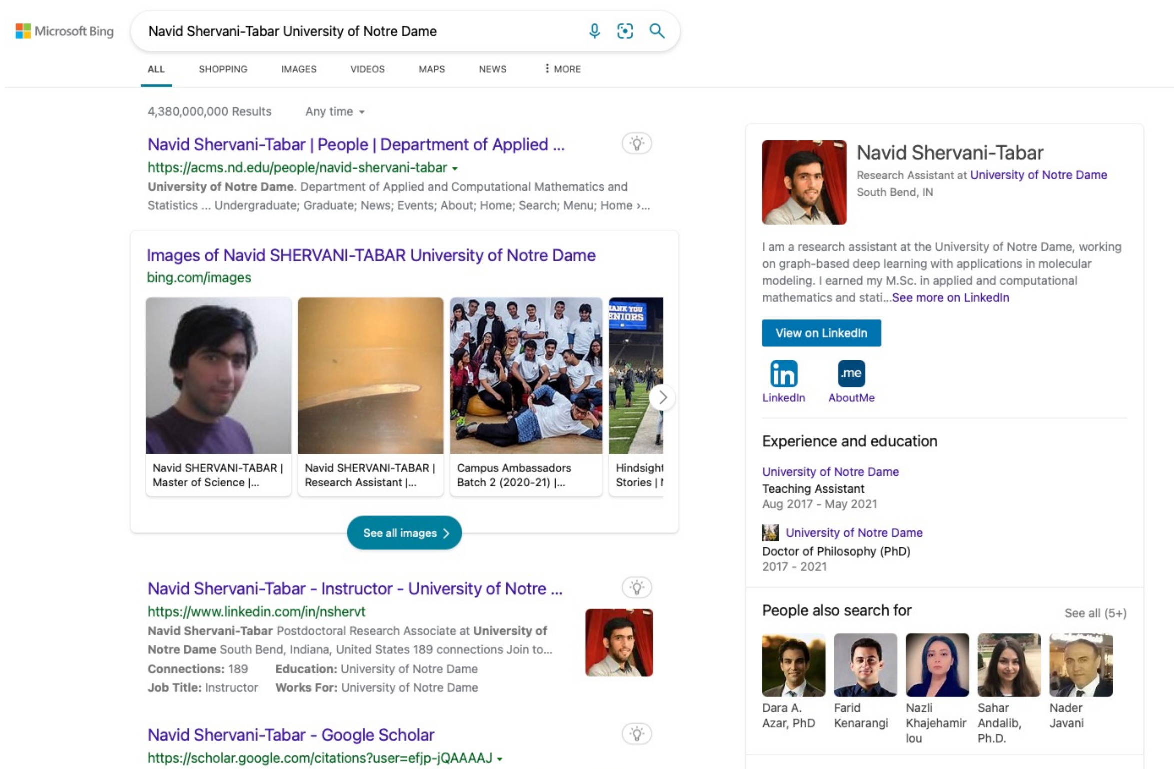
Knowledge Graph

- In recent years, a large number of Knowledge graphs have been created.
- DBpedia: Wikipedia
- Freebase: Human contributor
- Commerce: Amazon, Uber, Airbnb
- Finance: Bloomberg, Capital One.
- Social Media: LinkedIn
- AstraZeneca: Genomic research

Knowledge Graph

➤ Web Search

- Google knowledge graph
- Bing knowledge graph (Satori)



KG Completion

- Human knowledge about the relation between different entities is notoriously **incomplete**.
- The “place of birth” is **missing** for 71% of the people on the Freebase knowledge graph.
- As a result, knowledge graphs are often riddled with **missing edges**.

KG Completion

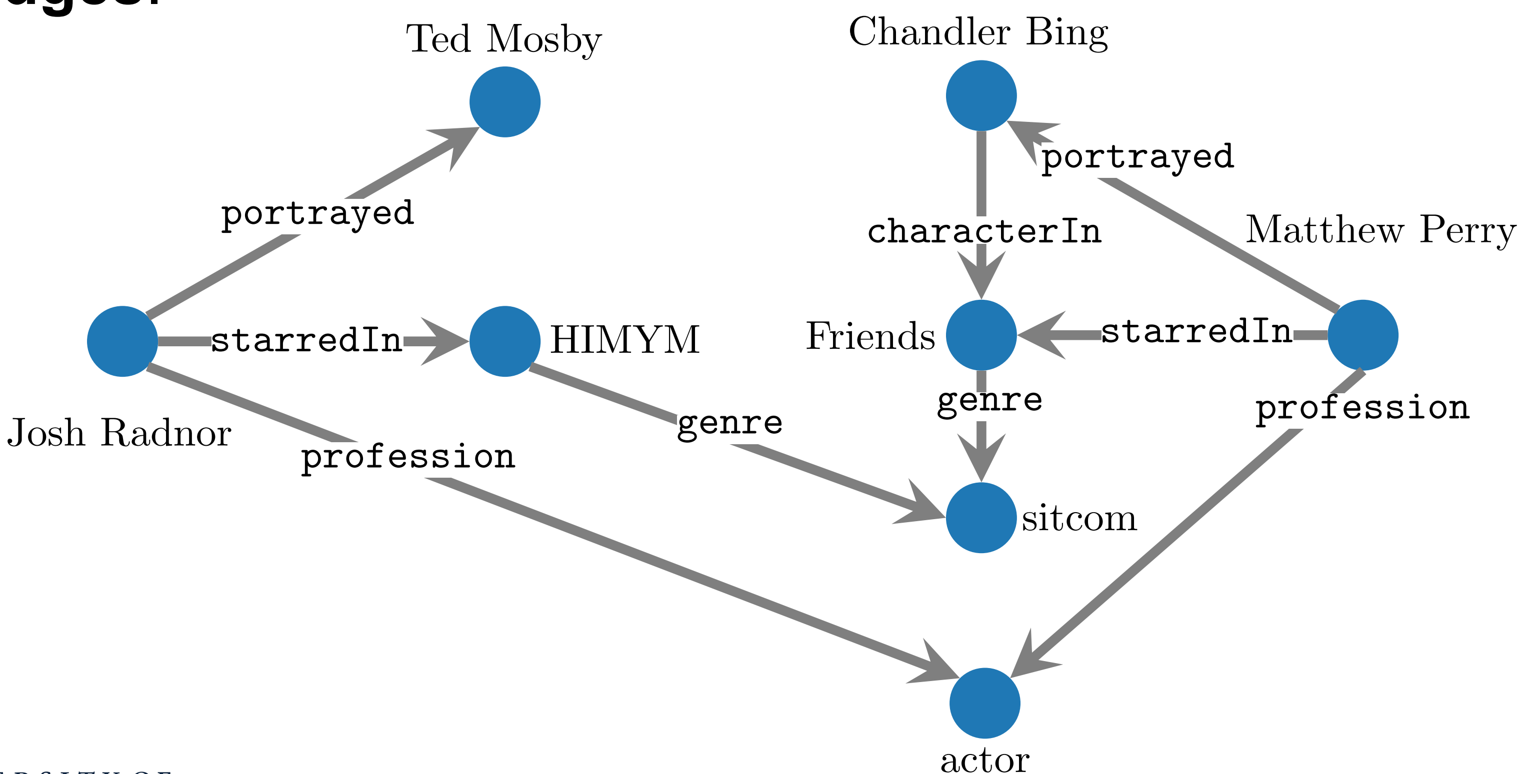
- Human knowledge about the relation between different entities is notoriously **incomplete**.
- The “place of birth” is **missing** for 71% of the people on the Freebase knowledge graph.
- As a result, knowledge graphs are often riddled with **missing edges**.
- **Knowledge graph completion** approaches are the set of techniques aiming to fill the missing edges.
- In addition, they can be used to **predict new facts** about the world.

KG Completion

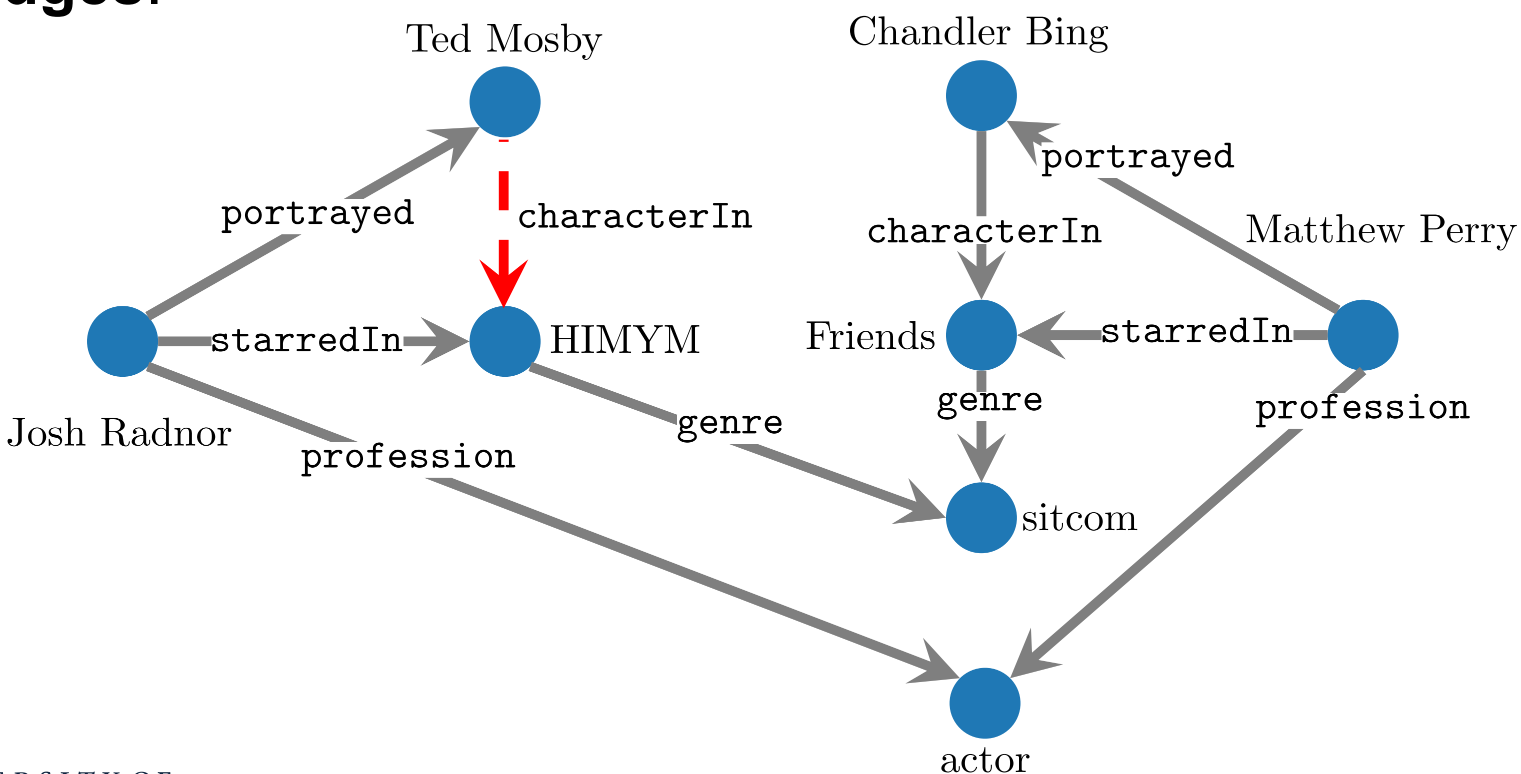
- In general, the knowledge graph completion involves **link prediction** problem, while in some cases we may deal with **node classification** as well.
- Link prediction refers to predicting existence of **missing edges**.
-

KG Completion

- In general, the knowledge graph completion involves **link prediction** problem, while in some cases we may deal with **node classification** as well.
- Link prediction refers to predicting existence of **missing edges**.



KG Completion

- In general, the knowledge graph completion involves **link prediction** problem, while in some cases we may deal with **node classification** as well.
- Link prediction refers to predicting existence of **missing edges**.
  - 

KG Embeddings

- In general, the knowledge graph completion involves **link prediction** problem, while in some cases we may deal with **node classification** as well.

- Link prediction refers to predicting existence of **missing edges**.

- To tackle this link prediction problem with machine learning models, we use **knowledge graph embedding** techniques to construct graph embeddings.

- These embeddings consist of **entity embeddings** $\boldsymbol{z}_h$ and $\boldsymbol{z}_t$ for nodes $v_i, v_j \in V$, and **relational embedding** r_τ that relates a pair of entities

Relational Patterns

- The **relations** represented on a knowledge graph follow different **logical patterns**.

Relational Patterns

- The **relations** represented on a knowledge graph follow different **logical patterns**.
- A relation τ is **symmetric** if

$$(v_h, \tau, v_t) \in E \Longleftrightarrow (v_t, \tau, v_h) \in E$$

-

Relational Patterns

- The **relations** represented on a knowledge graph follow different **logical patterns**.
- A relation τ is **symmetric** if

$$(v_h, \tau, v_t) \in E \iff (v_t, \tau, v_h) \in E$$

-

(Bill, *marriedTo*, Hillary) ⇔ (Hillary, *marriedTo*, Bill)

Relational Patterns

- The **relations** represented on a knowledge graph follow different **logical patterns**.
- A relation τ is **symmetric** if

$$(v_h, \tau, v_t) \in E \iff (v_t, \tau, v_h) \in E$$

- (Bill, *marriedTo*, Hillary) ⇔ (Hillary, *marriedTo*, Bill)

- An **anti-symmetric** relation τ indicates that

$$(v_h, \tau, v_t) \in E \implies (v_t, \tau, v_h) \notin E$$

Relational Patterns

- The **relations** represented on a knowledge graph follow different **logical patterns**.
- A relation τ is **symmetric** if

$$(v_h, \tau, v_t) \in E \iff (v_t, \tau, v_h) \in E$$

- (Bill, *marriedTo*, Hillary) ⇔ (Hillary, *marriedTo*, Bill)

- An **anti-symmetric** relation τ indicates that

$$(v_h, \tau, v_t) \in E \implies (v_t, \tau, v_h) \notin E$$

- A relation τ_1 is **inverse** to relation τ_2 if for all $v_h, v_t \in \mathrm{V}$.

$$(v_h, \tau_1, v_t) \in E \iff (v_t, \tau_2, v_h) \in E$$

Relational Patterns

- The **relations** represented on a knowledge graph follow different **logical patterns**.
- A relation τ is **symmetric** if

$$(v_h, \tau, v_t) \in E \iff (v_t, \tau, v_h) \in E$$

- (Bill, *marriedTo*, Hillary) ⇔ (Hillary, *marriedTo*, Bill)

- An **anti-symmetric** relation τ indicates that

$$(v_h, \tau, v_t) \in E \implies (v_t, \tau, v_h) \notin E$$

- A relation τ_1 is **inverse** to relation τ_2 if for all $v_h, v_t \in V$.

$$(v_h, \tau_1, v_t) \in E \iff (v_t, \tau_2, v_h) \in E$$

- (Jesus, *childOf*, Mary)⇔(Mary, *parentOf*, Jesus)

Relational Patterns

- **Compositionality**: Relation τ_3 is composed of relations τ_1 and τ_2 if for all $v_i, v_j, v_k \in \mathrm{V}$,

$$(v_i, \tau_1, v_j) \in E \ \wedge \ (v_j, \tau_2, v_k) \Longrightarrow (v_i, \tau_3, v_k) \in E$$

Relational Patterns

- **Compositionality**: Relation τ_3 is composed of relations τ_1 and τ_2 if for all $v_i, v_j, v_k \in \mathrm{V}$,

$$(v_i, \tau_1, v_j) \in E \ \wedge \ (v_j, \tau_2, v_k) \implies (v_i, \tau_3, v_k) \in E$$

  - College varsity teams draft players from students enrolled in that college.

(Ian Book, *draftedBy*, Fighting Irish)∧ (Fighting Irish, *school*, Notre Dame)

→ (Ian Book, *enrolledAt*, Notre Dame)

KG Embedding

- While in principle one can use GNN models to learn these embeddings, for simplicity, we are focusing on learning approaches based on **shallow embedding**.
- In the shallow embedding approach, we need to define a **loss**, a **similarity** measure, and a **decoder**.

KG Embedding

- While in principle one can use GNN models to learn these embeddings, for simplicity, we are focusing on learning approaches based on **shallow embedding**.
- In the shallow embedding approach, we need to define a **loss**, a **similarity** measure, and a **decoder**.
- Learning **node embeddings** involves reconstructing the local structure of a node v_i given its embedding $\boldsymbol{z}_i$.
- While learning node embeddings, we used various node-node **similarity measures** to define this **local structure**.

KG Embedding

- While in principle one can use GNN models to learn these embeddings, for simplicity, we are focusing on learning approaches based on **shallow embedding**.
- In the shallow embedding approach, we need to define a **loss**, a **similarity** measure, and a **decoder**.
- Learning **node embeddings** involves reconstructing the local structure of a node v_i given its embedding $\boldsymbol{z}_i$.
- While learning node embeddings, we used various node-node **similarity measures** to define this **local structure**.
- Similarly, in knowledge graph completion, we aim to **reconstruct the relation between nodes** v_i and v_j given the embeddings $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$.

KG Embedding

- While reconstructing node's neighborhood in simple graphs involves prediction of **existence of edges**, in knowledge graphs, we deal with **various edge types**.
- Therefore, we simply predict **multi-relational neighbors**.

KG Embedding

- While reconstructing node's neighborhood in simple graphs involves prediction of **existence of edges**, in knowledge graphs, we deal with **various edge types**.
- Therefore, we simply predict **multi-relational neighbors**.
- To accommodate reconstructing **multi-relational neighbors**, we augment the decoder with **relation specific embedding** $\boldsymbol{r}_\tau$.

KG Embedding

- While reconstructing node's neighborhood in simple graphs involves prediction of **existence of edges**, in knowledge graphs, we deal with **various edge types**.
- Therefore, we simply predict **multi-relational neighbors**.
- To accommodate reconstructing **multi-relational neighbors**, we augment the decoder with **relation specific embedding** $\boldsymbol{r}_\tau$.
- Therefore, we define the **decoder** as a map

$$f_d: R^d \times \mathcal{R} \times R^d \rightarrow R^+$$

to predict existence of edge types between nodes v_i and v_j.

KG Embedding

- While reconstructing node's neighborhood in simple graphs involves prediction of **existence of edges**, in knowledge graphs, we deal with **various edge types**.
- Therefore, we simply predict **multi-relational neighbors**.
- To accommodate reconstructing **multi-relational neighbors**, we augment the decoder with **relation specific embedding** $\boldsymbol{r}_\tau$.
- Therefore, we define the **decoder** as a map

$$f_d: R^d \times \mathcal{R} \times R^d \rightarrow R^+$$

  to predict existence of edge types between nodes v_i and v_j.
- Given node embeddings $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ and relational embedding $\boldsymbol{r}_\tau$, a decoder f_d evaluates the plausibility of the edge type τ between nodes v_i and v_j.

Multi-relational Decoder

- When **training** the model for the link prediction task, we predict the **plausibility** of the relational data. These relations have been extracted from the external sources.
- Then, we used the trained model to predict the plausibility of existence of **missing edges**.

Multi-relational Decoder

- When **training** the model for the link prediction task, we predict the **plausibility** of the relational data. These relations have been extracted from the external sources.
- Then, we used the trained model to predict the plausibility of existence of **missing edges**.
- The **decoders** can generally be divided into two categories.

Multi-relational Decoder

- When **training** the model for the link prediction task, we predict the **plausibility** of the relational data. These relations have been extracted from the external sources.
- Then, we used the trained model to predict the plausibility of existence of **missing edges**.
- The **decoders** can generally be divided into two categories.
  - **Translational decoders**
    - These approaches derive the likelihood of a relation between two entities from the **distance** between their embeddings.

Multi-relational Decoder

- When **training** the model for the link prediction task, we predict the **plausibility** of the relational data. These relations have been extracted from the external sources.
- Then, we used the trained model to predict the plausibility of existence of **missing edges**.
- The **decoders** can generally be divided into two categories.
  - **Translational decoders**
    - These approaches derive the likelihood of a relation between two entities from the **distance** between their embeddings.
  - **Tensor decomposition decoders**

Translational Models

- In **translational** models, a **distance** function d measures the closeness of the two entities.

Translational Models

- In **translational** models, a **distance** function d measures the closeness of the two entities.
- In the translational models, **relational embeddings** represent **translations** in the embedding space.
- In other words, we assume that the **relational** embedding $\boldsymbol{r}_\tau$ translates **head node** embedding $\boldsymbol{z}_h$ to **tail node** embedding $\boldsymbol{z}_t$.

Translational Models

- In **translational** models, a **distance** function d measures the closeness of the two entities.
- In the translational models, **relational embeddings** represent **translations** in the embedding space.
- In other words, we assume that the **relational** embedding $\boldsymbol{r}_\tau$ translates **head node** embedding $\boldsymbol{z}_h$ to **tail node** embedding $\boldsymbol{z}_t$.
- This **closeness** quantifies the probability of the two entities being in a **relationship**.

$$f_d\left(\boldsymbol{z}_h, \tau, \boldsymbol{z}_t\right) = -d\left(\boldsymbol{z}_h + \boldsymbol{r}_\tau, \boldsymbol{z}_t\right)$$

TransE

- The most basic translational model is called **TransE**.
- Given a tuple (v_h, τ, v_t) representing a fact, TransE learns entity and relational embeddings $\boldsymbol{z}_h$, $\boldsymbol{z}_t$, and $\boldsymbol{r}_\tau$ so as to minimize the distance between $\boldsymbol{z}_h + \boldsymbol{r}_\tau$ and $\boldsymbol{z}_t$.
- TransE **scores** (v_h, τ, v_t) based on closeness of $\boldsymbol{z}_h + \boldsymbol{r}_\tau$ and $\boldsymbol{z}_t$.

TransE

- The most basic translational model is called **TransE**.
- Given a tuple (v_h, τ, v_t) representing a fact, TransE learns entity and relational embeddings $\boldsymbol{z}_h$, $\boldsymbol{z}_t$, and $\boldsymbol{r}_\tau$ so as to minimize the distance between $\boldsymbol{z}_h + \boldsymbol{r}_\tau$ and $\boldsymbol{z}_t$.
- TransE **scores** (v_h, τ, v_t) based on closeness of $\boldsymbol{z}_h + \boldsymbol{r}_\tau$ and $\boldsymbol{z}_t$.
- Mathematically put, TransE defines the decoder as

$$f_d\left(\boldsymbol{z}_h, \tau, \boldsymbol{z}_t\right) = -\left\|\boldsymbol{z}_h + \boldsymbol{r}_\tau - \boldsymbol{z}_t\right\|$$

  where $\boldsymbol{r}_\tau \in \mathbb{R}^d$ is the relational embedding and $\boldsymbol{z}_h, \boldsymbol{z}_t \in \mathbb{R}^d$ are entity embeddings.
- Therefore, the distance between the translated entity embedding $\boldsymbol{z}_h + \boldsymbol{r}_\tau$ and the entity embedding $\boldsymbol{z}_t$ represents the **likelihood** of **existence** of relation type τ between v_h and v_t.

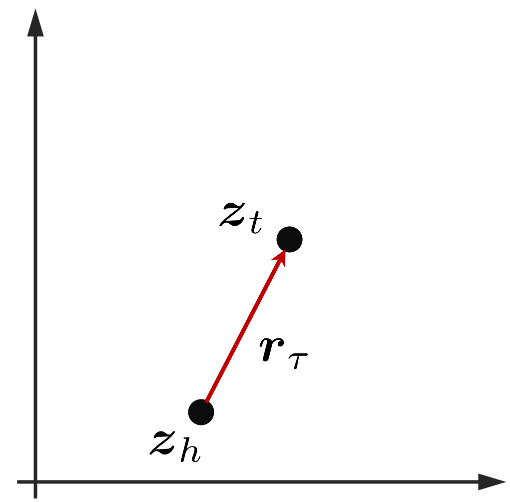
TransE: Relational Patterns

- **Symmetry:**

$$\boldsymbol{z}_h + \boldsymbol{r}_\tau = \boldsymbol{z}_t$$
$$\boldsymbol{z}_t + \boldsymbol{r}_\tau = \boldsymbol{z}_h$$

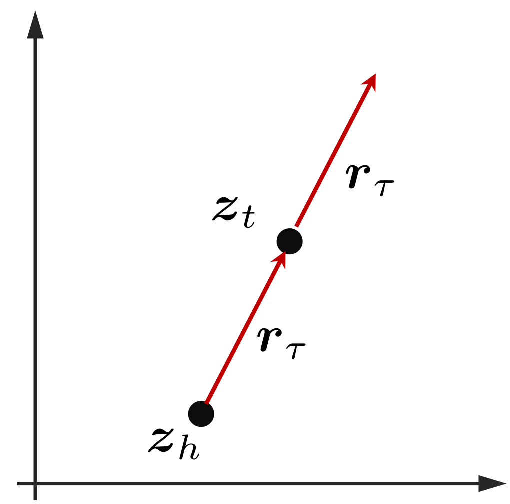
$$\implies \quad \boldsymbol{r}_\tau = 0, \quad \boldsymbol{z}_h = \boldsymbol{z}_t$$

Cannot.



- **Anti-Symmetry**:

$$\boldsymbol{z}_h + \boldsymbol{r}_\tau = \boldsymbol{z}_t$$
$$\boldsymbol{z}_t + \boldsymbol{r}_\tau \neq \boldsymbol{z}_h$$

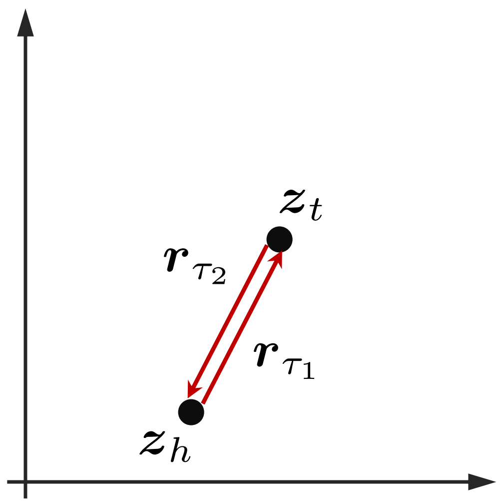


TransE: Relational Patterns

- Inverse:

$$\boldsymbol{z}_h + \boldsymbol{r}_{\tau_1} = \boldsymbol{z}_t$$
$$\boldsymbol{z}_t + \boldsymbol{r}_{\tau_2} = \boldsymbol{z}_h$$

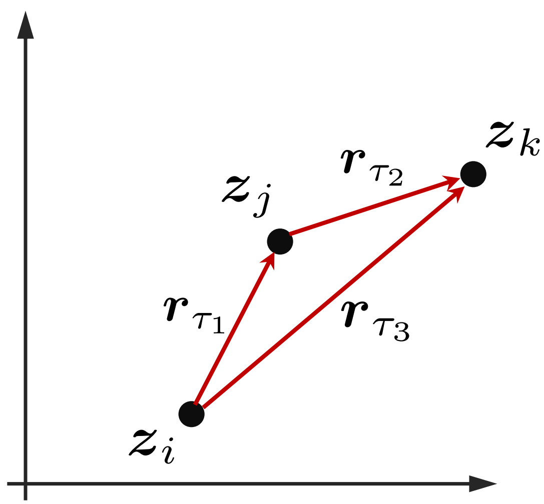
$$\implies \quad \boldsymbol{r}_{\tau_1} = -\boldsymbol{r}_{\tau_2}$$



- Compositionality:

$$\boldsymbol{z}_i + \boldsymbol{r}_{\tau_1} = \boldsymbol{z}_j$$
$$\boldsymbol{z}_j + \boldsymbol{r}_{\tau_2} = \boldsymbol{z}_k$$
$$\boldsymbol{z}_i + \boldsymbol{r}_{\tau_3} = \boldsymbol{z}_k$$

$$\implies \quad \boldsymbol{r}_{\tau_1} + \boldsymbol{r}_{\tau_2} = \boldsymbol{r}_{\tau_3}$$



TransR

- To address these issues, many **variants** of the TransE model have been proposed.

TransR

- To address these issues, many **variants** of the TransE model have been proposed.
- **TransR** is defined based on the intuition that, entities have **various aspects**, and different relational spaces should represent different aspect of these entities.

TransR

- To address these issues, many **variants** of the TransE model have been proposed.
- **TransR** is defined based on the intuition that, entities have **various aspects**, and different relational spaces should represent different aspect of these entities.
- TransR approach represents each relation type τ in a **distinct vector space**.

$$\boldsymbol{z}_h^{\tau} = \mathbf{M}_{\tau}\boldsymbol{z}_h, \qquad \boldsymbol{z}_t^{\tau} = \mathbf{M}_{\tau}\boldsymbol{z}_t$$

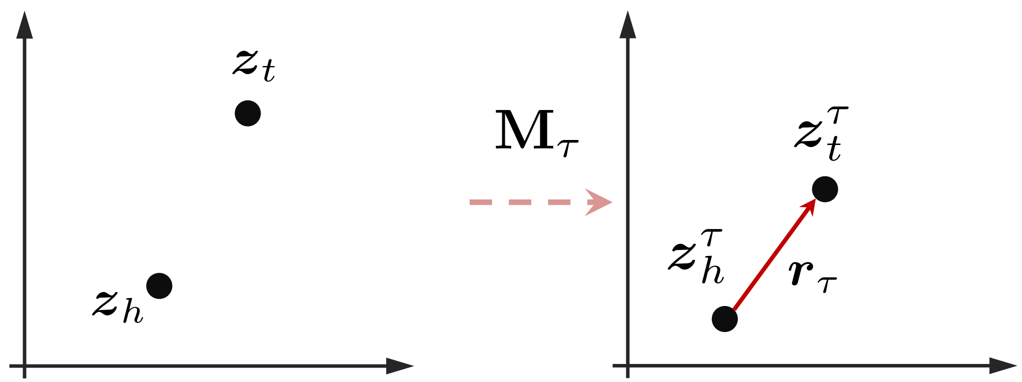
- TransR uses relation-specific **projection** matrices M_τ to project node embeddings onto the relation-specific vector space before **translating** them.

TransR

- We can formulate the TransR model as

$$f_d\left(\boldsymbol{z}_h, \tau, \boldsymbol{z}_t\right) = -\left\|\mathbf{M}_\tau \boldsymbol{z}_h + \boldsymbol{r}_\tau - \mathbf{M}_\tau \boldsymbol{z}_t\right\|$$

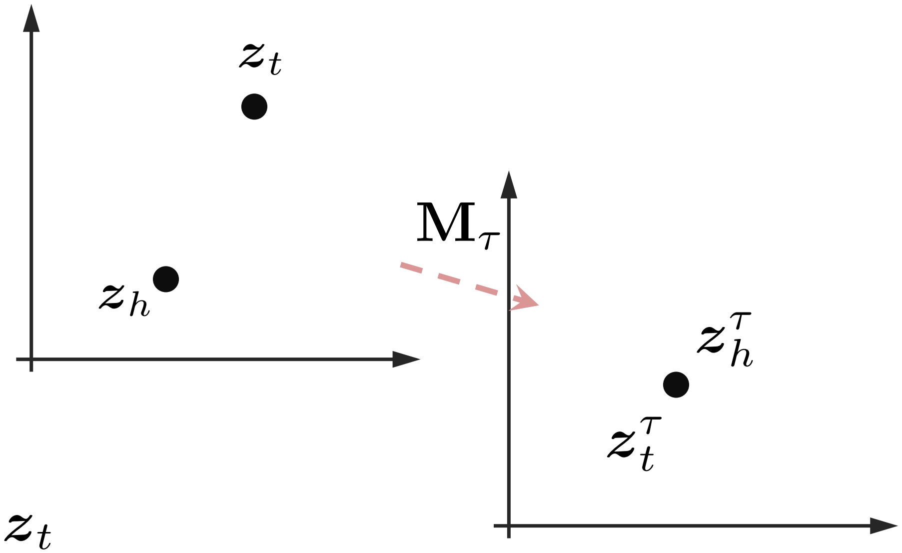
where $\boldsymbol{z}_h, \boldsymbol{z}_t \in \mathbb{R}^d$, $\boldsymbol{r}_\tau \in \mathbb{R}^k$ and $\mathbf{M}_\tau \in \mathbb{R}^{k\times d}$.



TransR: Relational Patterns

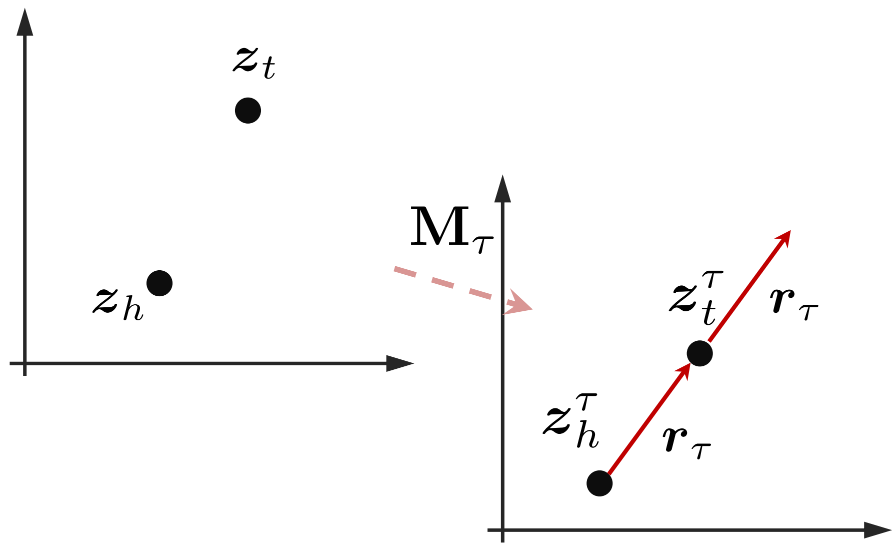
- **Symmetry:**

$$\mathbf{M}_\tau \boldsymbol{z}_h + \boldsymbol{r}_\tau = \mathbf{M}_\tau \boldsymbol{z}_t$$
$$\mathbf{M}_\tau \boldsymbol{z}_t + \boldsymbol{r}_\tau = \mathbf{M}_\tau \boldsymbol{z}_h$$
$$\Longrightarrow \quad \boldsymbol{r}_\tau = 0$$
$$\mathbf{M}_\tau \boldsymbol{z}_h = \mathbf{M}_\tau \boldsymbol{z}_t$$



- **Anti-Symmetry:**

$$\mathbf{M}_\tau \boldsymbol{z}_h + \boldsymbol{r}_\tau = \mathbf{M}_\tau \boldsymbol{z}_t$$
$$\mathbf{M}_\tau \boldsymbol{z}_t + \boldsymbol{r}_\tau \neq \mathbf{M}_\tau \boldsymbol{z}_h$$



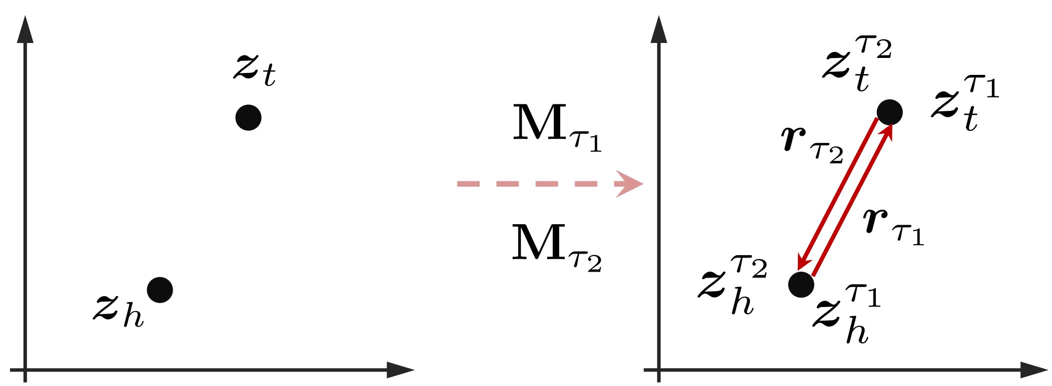
TransR: Relational Patterns

- Inverse:

$$\mathbf{M}_{\tau_1}\boldsymbol{z}_h + \boldsymbol{r}_{\tau_1} = \mathbf{M}_{\tau_1}\boldsymbol{z}_t$$
$$\mathbf{M}_{\tau_2}\boldsymbol{z}_t + \boldsymbol{r}_{\tau_2} = \mathbf{M}_{\tau_2}\boldsymbol{z}_h$$

- If $M_{\tau_1} = M_{\tau_2}$, we can project entity embeddings to the same relation specific space, and we can use

$$\boldsymbol{r}_{\tau_1} = -\boldsymbol{r}_{\tau_2}$$



TransR: Relational Patterns

➢ **Compositionality:**

$$\mathbf{M}_{\tau_1}\boldsymbol{z}_i + \boldsymbol{r}_{\tau_1} = \mathbf{M}_{\tau_1}\boldsymbol{z}_j$$
$$\mathbf{M}_{\tau_2}\boldsymbol{z}_j + \boldsymbol{r}_{\tau_2} = \mathbf{M}_{\tau_2}\boldsymbol{z}_k$$
$$\mathbf{M}_{\tau_3}\boldsymbol{z}_i + \boldsymbol{r}_{\tau_3} = \mathbf{M}_{\tau_3}\boldsymbol{z}_k$$

❖ Since translations between relation specific spaces are not defined, we can't represent compositional patterns using TransR.

RotatE

- RotatE, learns translational embeddings in the **complex space**.
- In this method, relations are modeled as **element-wise rotations** in the complex vector space.
- Based on Euler's identity

$$e^{i\theta} = \cos\theta + i\sin\theta$$

A unitary complex number can represent a relation in the complex plane.

- Motivated by Euler's identity, RotatE defines the decoder as

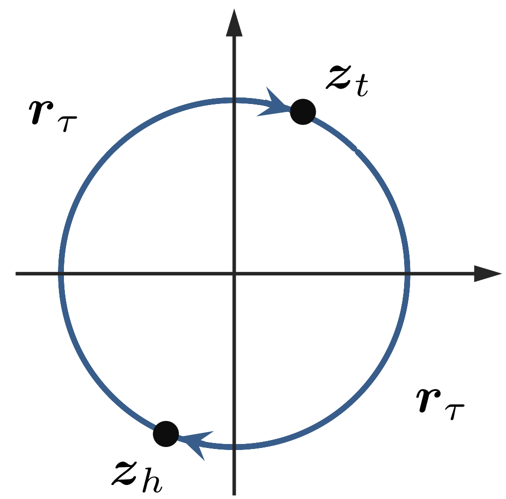
$$f_d\left(\boldsymbol{z}_h, \tau, \boldsymbol{z}_t\right) = -\left\|\boldsymbol{z}_h \odot \boldsymbol{r}_\tau - \boldsymbol{z}_t\right\|$$

where $\boldsymbol{z}_h, \boldsymbol{z}_t, \boldsymbol{r}_\tau \in \mathbb{C}^d$ and $|[\boldsymbol{r}_\tau]_i| = 1$.

RotatE: Relational Patterns

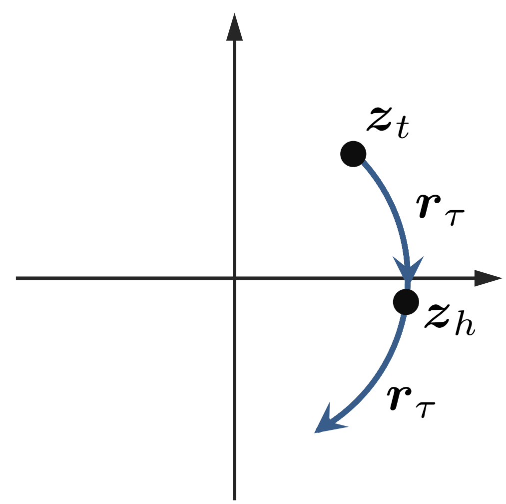
- Symmetry:

$$\boldsymbol{z}_h \odot \boldsymbol{r}_\tau = \boldsymbol{z}_t$$
$$\boldsymbol{z}_t \odot \boldsymbol{r}_\tau = \boldsymbol{z}_h$$
$$\Longrightarrow \quad [\boldsymbol{r}_\tau]_j = -1$$



- Anti-Symmetry:

$$\boldsymbol{z}_h \odot \boldsymbol{r}_\tau = \boldsymbol{z}_t$$
$$\boldsymbol{z}_t \odot \boldsymbol{r}_\tau \neq \boldsymbol{z}_h$$



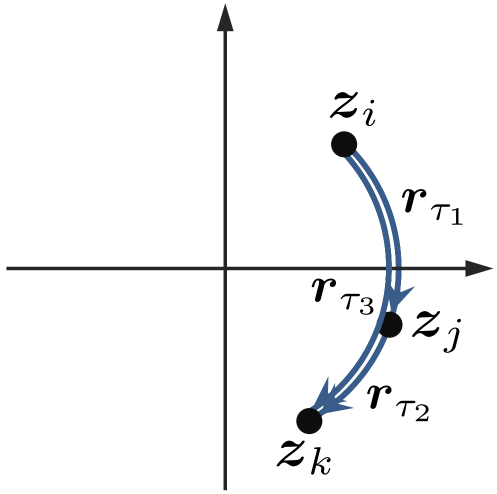
RotatE: Relational Patterns

- Inverse:

$$\boldsymbol{r}_{\tau_2} = \bar{\boldsymbol{r}}_{\tau_1}$$

- Compositionality:

$$\theta_3 = \theta_2 + \theta_1$$



Tensor Decomposition Models

- Another group of decoding models are tensor decomposition models.
- Here we review the following methods
  - RESCAL
  - DistMult
  - ComplEx

RESCAL

- RESCAL is one of the earliest learning approaches for multi-relational embeddings.
- RESCAL defines a decoder based on **pair-wise interaction** of the embeddings

$$f_d\left(\boldsymbol{z}_h, \tau, \boldsymbol{z}_t\right) = \boldsymbol{z}_h^T \boldsymbol{r}_\tau \boldsymbol{z}_t$$

  where $\boldsymbol{r}_\tau \in \mathbb{R}^{d \times d}$ is a learnable **relation-specific matrix** and $\boldsymbol{z}_h, \boldsymbol{z}_t \in \mathbb{R}^d$ are entity embeddings.

- Element $[\boldsymbol{r}_\tau]_{i,j}$ indicates how much embedding dimensions $[\boldsymbol{z}_h]_i$ and $[\boldsymbol{z}_t]_j$ interact in the relation type τ.

RESCAL

- RESCAL is one of the earliest learning approaches for multi-relational embeddings.
- RESCAL defines a decoder based on **pair-wise interaction** of the embeddings

$$f_d(\boldsymbol{z}_h, \tau, \boldsymbol{z}_t) = \boldsymbol{z}_h^T \boldsymbol{r}_\tau \boldsymbol{z}_t$$

  where $\boldsymbol{r}_\tau \in \mathbb{R}^{d \times d}$ is a learnable **relation-specific matrix** and $\boldsymbol{z}_h, \boldsymbol{z}_t \in \mathbb{R}^d$ are entity embeddings.

- Element $[\boldsymbol{r}_\tau]_{i,j}$ indicates how much embedding dimensions $[\boldsymbol{z}_h]_i$ and $[\boldsymbol{z}_t]_j$ interact in the relation type τ.
- While RESCAL can infer the discussed relational patterns, it is **computationally expensive**.

DistMult

- DistMult method defines the decoder f_d using element-wise product between head embedding $\boldsymbol{z}_h$, relational embedding $\boldsymbol{r}_\tau$ and the tail embedding $\boldsymbol{z}_t$

$$f_d\left(\boldsymbol{z}_h, \tau, \boldsymbol{z}_t\right) = \sum_{i=1}^{d} [\boldsymbol{z}_h]_i [\boldsymbol{r}_\tau]_i [\boldsymbol{z}_t]_i$$

  where $\boldsymbol{z}_h, \boldsymbol{z}_t, \boldsymbol{r}_\tau \in \mathbb{R}^d$.

- This type of decoder is a generalization of the dot-product decoders.

DistMult: Relational Patterns

- **Symmetry:**

$$f_d(\boldsymbol{z}_h, \tau, \boldsymbol{z}_t) = \sum_{i=1}^{d} [\boldsymbol{z}_h]_i [\boldsymbol{r}_\tau]_i [\boldsymbol{z}_t]_i$$
$$= \sum_{i=1}^{d} [\boldsymbol{z}_t]_i [\boldsymbol{r}_\tau]_i [\boldsymbol{z}_h]_i$$
$$= f_d(\boldsymbol{z}_t, \tau, \boldsymbol{z}_h)$$

Therefore,

$$(v_h, \tau, v_t) \in E \iff (v_t, \tau, v_h) \in E$$

- **Anti-Symmetry:**

  - Since f_d is commutative, DistMult can model symmetric relational patters, but **not** anti-symmetric patterns.

DistMult: Relational Patterns

- **Inverse:**
  - For two node embeddings with inverse relations,

$$\sum_{i=1}^{d} [\boldsymbol{z}_h]_i \, [\boldsymbol{r}_{\tau_1}]_i \, [\boldsymbol{z}_t]_i = \sum_{i=1}^{d} [\boldsymbol{z}_t]_i \, [\boldsymbol{r}_{\tau_2}]_i \, [\boldsymbol{z}_h]_i$$

    Therefore, we conclude that

$$\boldsymbol{r}_{\tau_1} = \boldsymbol{r}_{\tau_2}$$

  - This is not acceptable as two different relations are represented with the same embedding.

ComplEx

- ComplEx approach improves DistMult by generalizing entity and relational embeddings to the **complex domain**.
- The ComplEx model is formulated as

$$f_d\left(\boldsymbol{z}_h, \tau, \boldsymbol{z}_t\right) = \mathrm{Re}\left(\sum_{i=1}^{d} [\boldsymbol{z}_h]_i [\boldsymbol{r}_\tau]_i [\bar{\boldsymbol{z}}_t]_i\right)$$

  where $\boldsymbol{z}_h, \boldsymbol{r}_\tau, \boldsymbol{z}_t \in \mathbb{C}^d$ and $\bar{\boldsymbol{z}}_t$represent the complex conjugate of $\boldsymbol{z}_t$.
- Using the **complex conjugate** for entity v_t facilities modeling **asymmetric** relations.

ComplEx: Relational Patterns

- Symmetry:

$$d_f(\boldsymbol{z}_h, \tau, \boldsymbol{z}_t) = \text{Re}\left(\sum_{i=1}^{d} [\boldsymbol{z}_h]_i [\boldsymbol{r}_\tau]_i [\bar{\boldsymbol{z}}_t]_i\right)$$

$$= \sum_{i=1}^{d} \text{Re}\left([\boldsymbol{z}_h]_i [\boldsymbol{r}_\tau]_i [\bar{\boldsymbol{z}}_t]_i\right)$$

- If $Im(\boldsymbol{r}_\tau) = 0$, then

$$= \sum_{i=1}^{d} [\boldsymbol{r}_\tau]_i \text{Re}\left([\boldsymbol{z}_h]_i [\bar{\boldsymbol{z}}_t]_i\right)$$

$$= \sum_{i=1}^{d} [\boldsymbol{r}_\tau]_i \text{Re}\left([\bar{\boldsymbol{z}}_h]_i [\boldsymbol{z}_t]_i\right)$$

$$= \text{Re}\left(\sum_{i=1}^{d} [\boldsymbol{z}_t]_i [\boldsymbol{r}_\tau]_i [\bar{\boldsymbol{z}}_h]_i\right)$$

$$= d_f(\boldsymbol{z}_t, \tau, \boldsymbol{z}_h)$$

ComplEx: Relational Patterns

- **Inverse:**
- By setting

$$r_{\tau_1} = \bar{r}_{\tau_2}$$

It can represent inverse relations

- **Compositionality:**
- It can't represent compositional relations

Summary

- Knowledge graphs
- Knowledge graph Completion
- Multi-relational decoders
  - TransE
  - TransR
  - RotatE
  - RESCAL
  - DistMult
  - ComplEx