Heterogeneous Graphs and Knowledge Graph Embeddings

Prof. O-Joun Lee

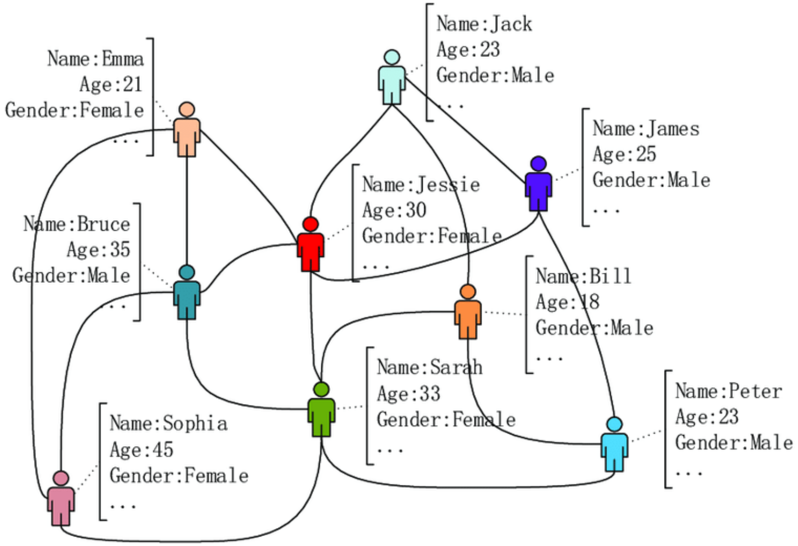
Dept. of Artificial Intelligence,
The Catholic University of Korea
ojlee@catholic.ac.kr

네트워크 과학 연구실 NETWORK SCIENCE LAB
가톨릭대학교 THE CATHOLIC UNIVERSITY OF KOREA

Contents

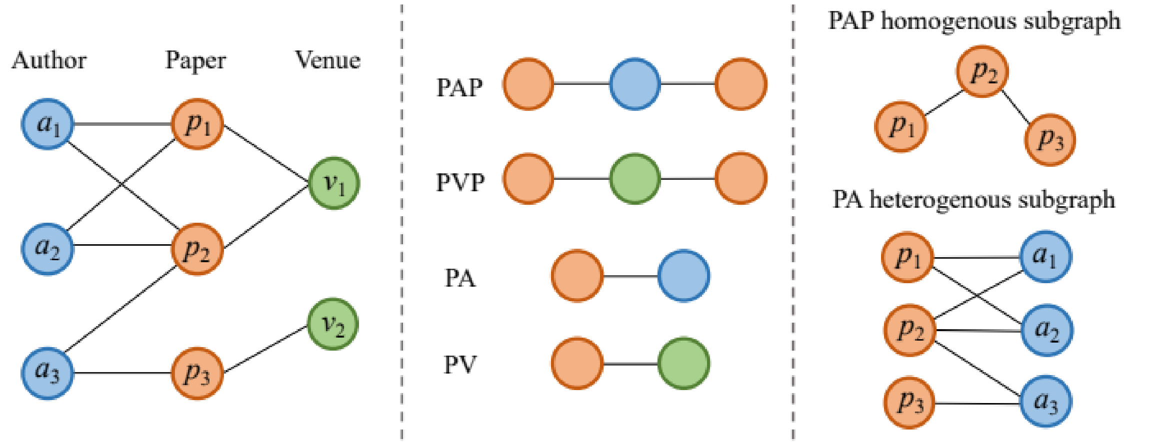
From Homogeneous to Heterogeneous Graphs

- Many real-world systems, such as molecular graphs, social networks, and knowledge graphs, involve multiple entities and relations.
- For example: In Social Networks
  - Entities (Nodes): Users, posts, or hashtags.
  - Relations (Edges): Friendships, likes, shares, or mentions.
  - Reason for Heterogeneity: Interactions vary in type and significance, and users/posts exhibit diverse roles in the network.



Heterogeneous Graphs

- **Objective:**
  - So far we only handle graphs with one edge type.
  - Question: How to handle (directed) graphs with multiple edge types (heterogeneous graphs)?
  - For example:
    - a) heterogeneous graph with three types of nodes (i.e., authors, papers, venues).
    - (b) Four metapaths: PAP, PVP, PA, and PV.
    - (c) The metapath-based homogeneous graph and heterogeneous graph



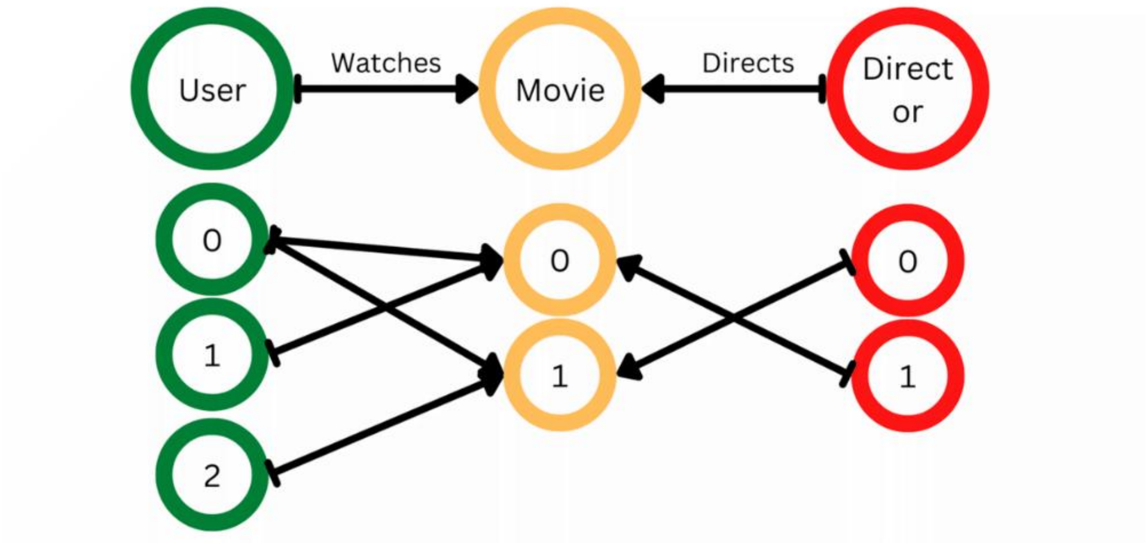
(a) Heterogeneous Graph (b) Metapaths (c) Metapath-based Graphs

Heterogeneous Graphs

- A heterogeneous graph is defined as:

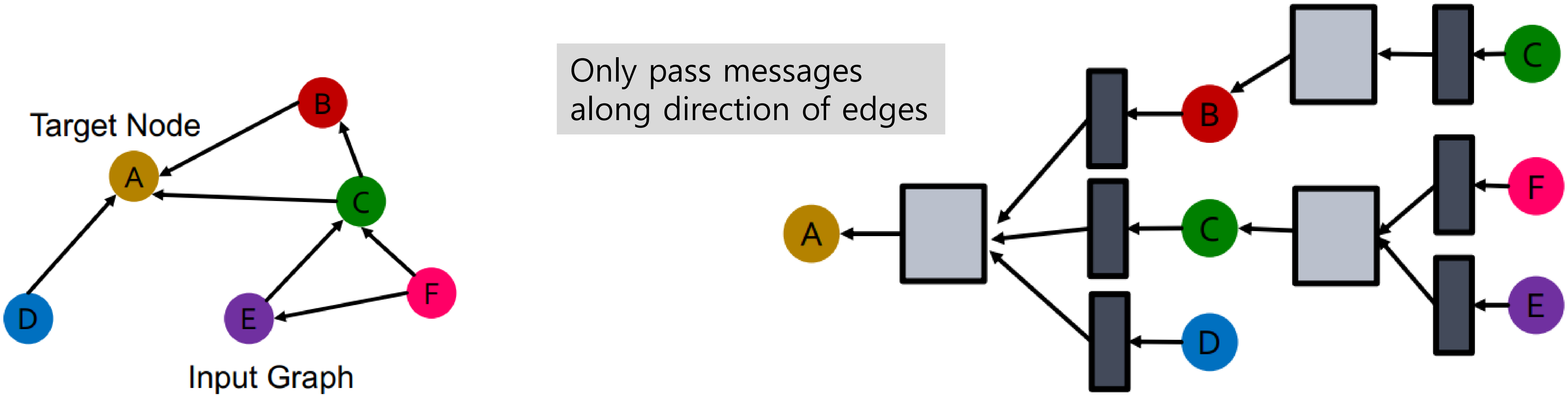
$$\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{R}, \boldsymbol{T})$$

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



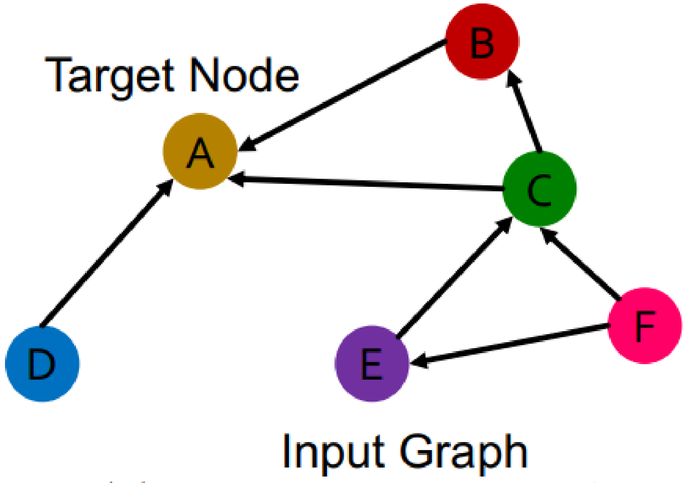
Heterogeneous Graphs: Relational GCN

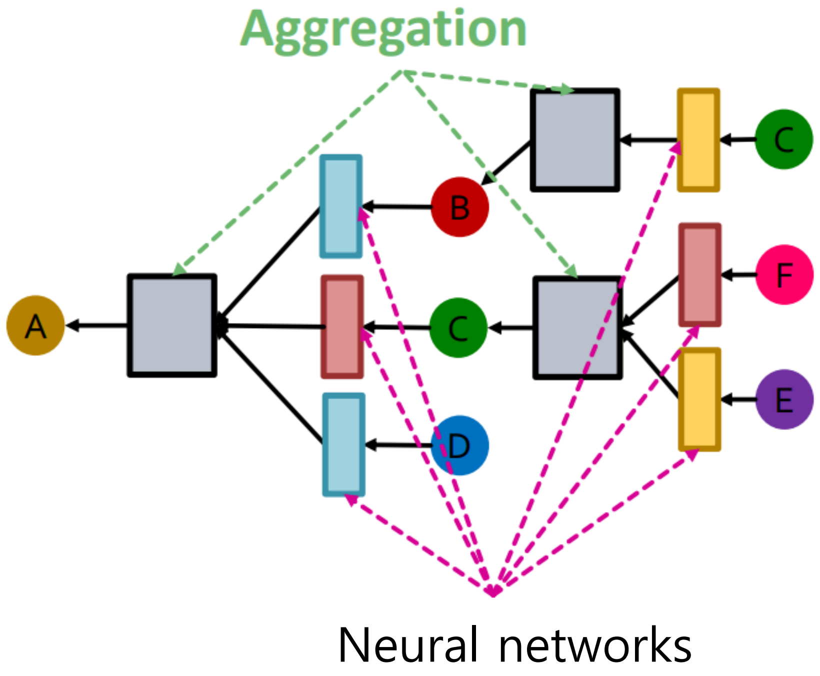
- We will extend GCN to handle heterogeneous graphs with multiple edge/relation types
- We start with a directed graph with one relation
  - How do we run GCN and update the representation of the target node A on this graph?



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

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





- 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 & Self-loop::

$$\mathbf{m}_{u,r}^{(l)} = \frac{1}{c_{v,r}}\mathbf{W}_r^{(l)}\mathbf{h}_u^{(l)} \qquad \mathbf{m}_v^{(l)} = \mathbf{W}_0^{(l)}\mathbf{h}_v^{(l)}$$

  - **Aggregation**: Sum over messages from neighbors and self-loop, then apply activation

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

Relational GCN vs. GCN

- How to define Message + Aggregation?

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

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

- Aggregation:

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

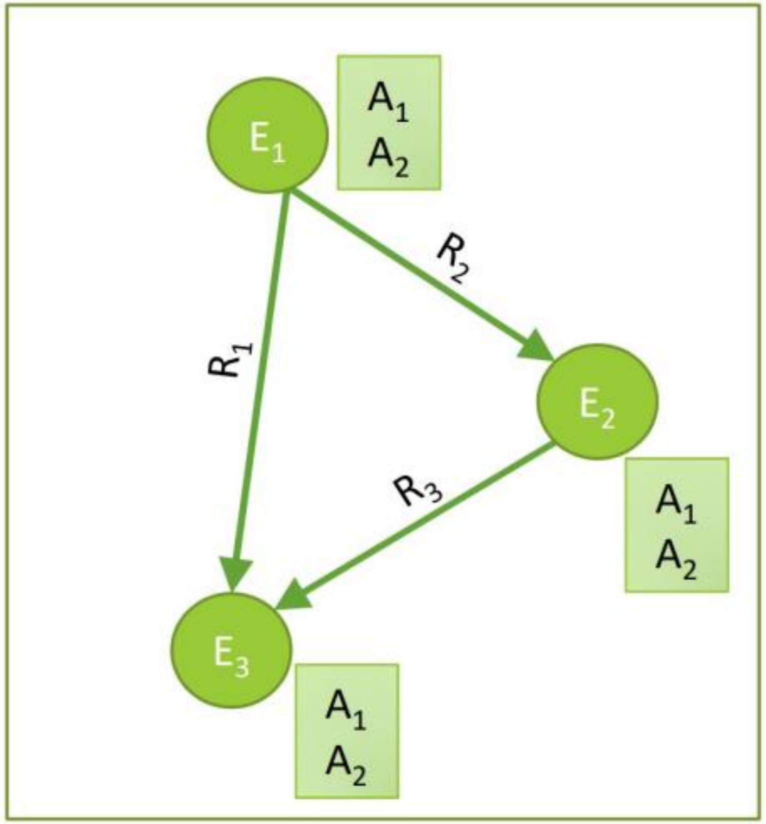
Message

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

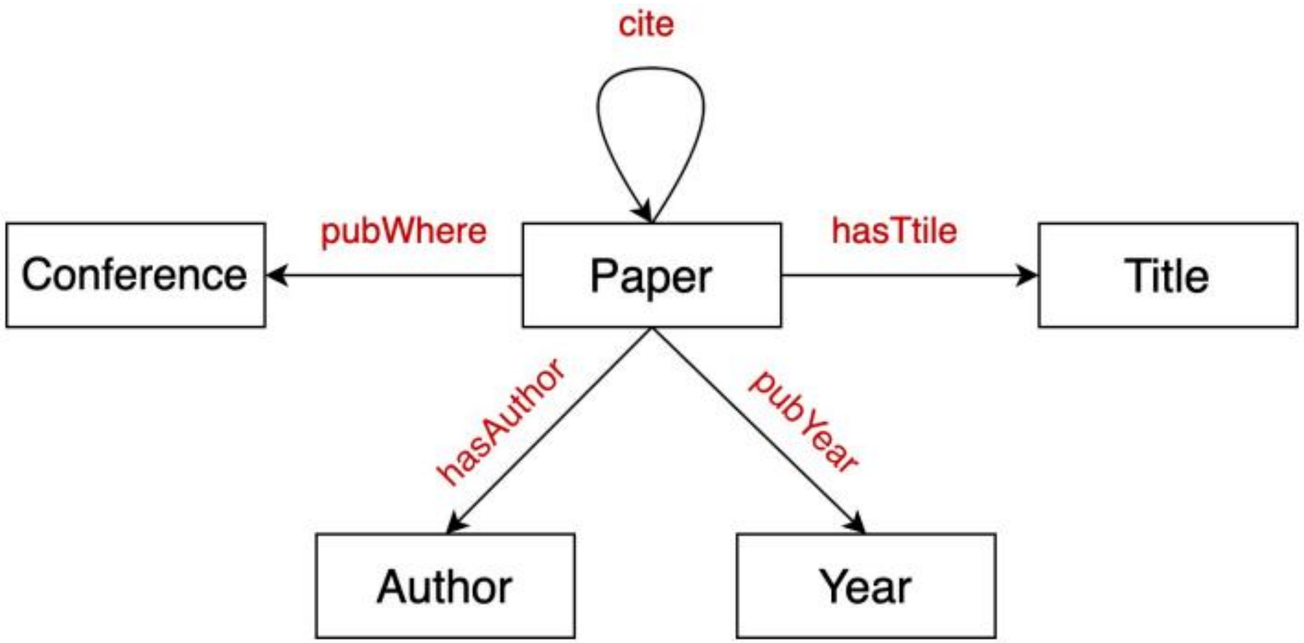
Relational GCN

GCN

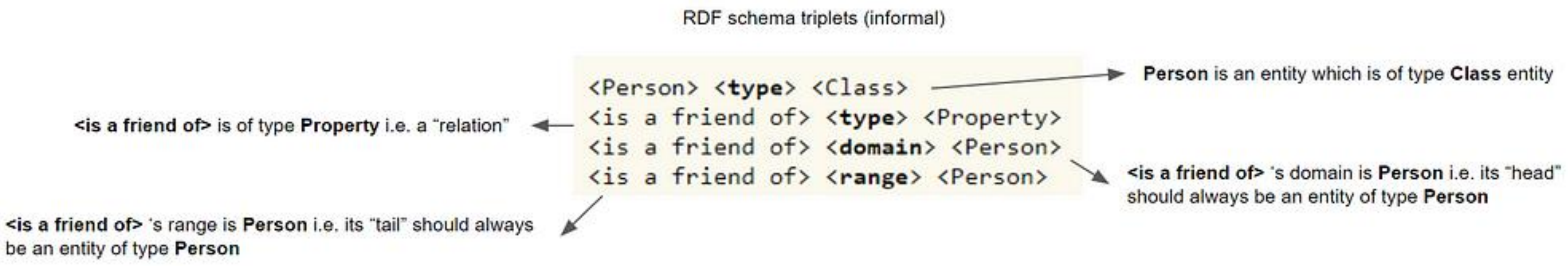
- Knowledge in graph form:
  - Capture entities, types, and relationships
  - Nodes are entities
  - Nodes are labeled with their types
  - Edges between two nodes capture relationships between entities
- A Knowledge Graph (KG) is indeed a prime example of a heterogeneous graph.



- An example of Bibliographic Networks
  - Node types: paper, title, author, conference, year
  - Relation types: pubWhere, pubYear, hasTitle, hasAuthor, cite

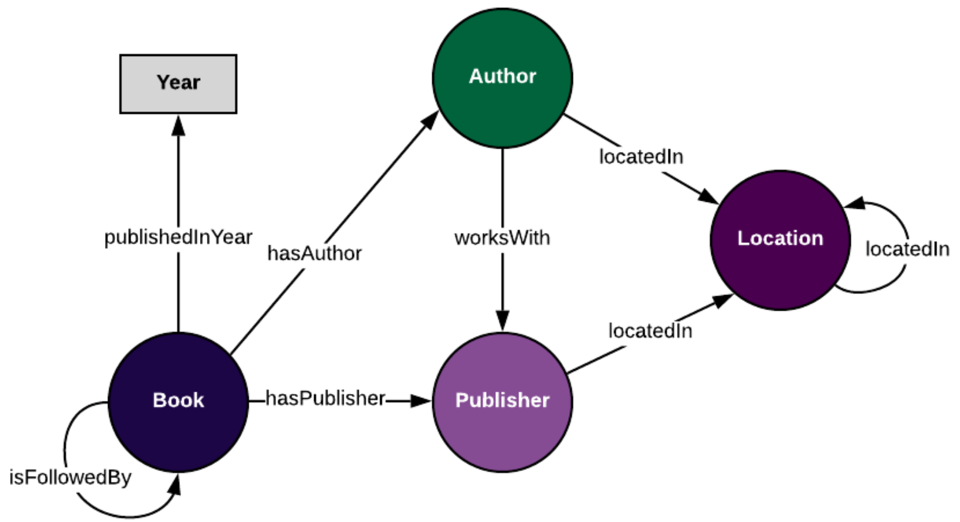


- **Knowledge graph Ontology**
  - An ontology is a model of the world (practically only a subset), listing the types of entities, the relationships that connect them, and constraints on the ways that entities and relationships can be combined.
  - Resource Description Framework (RDF) and Web Ontology Language (OWL) are some of the vocabulary frameworks used to model ontology.



Why we need Ontologies?

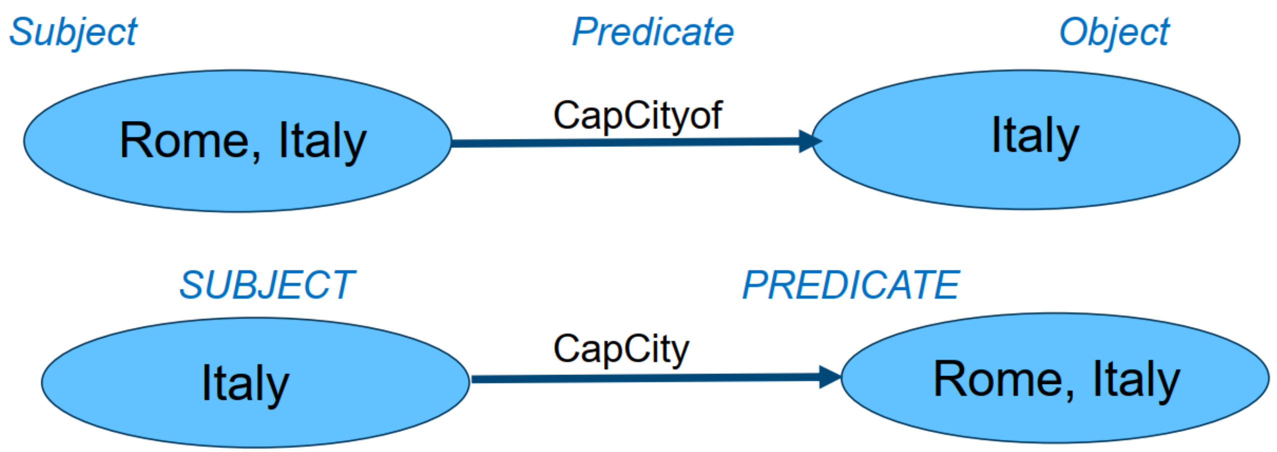
- To share common understanding of the structure of information among people or objects
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge



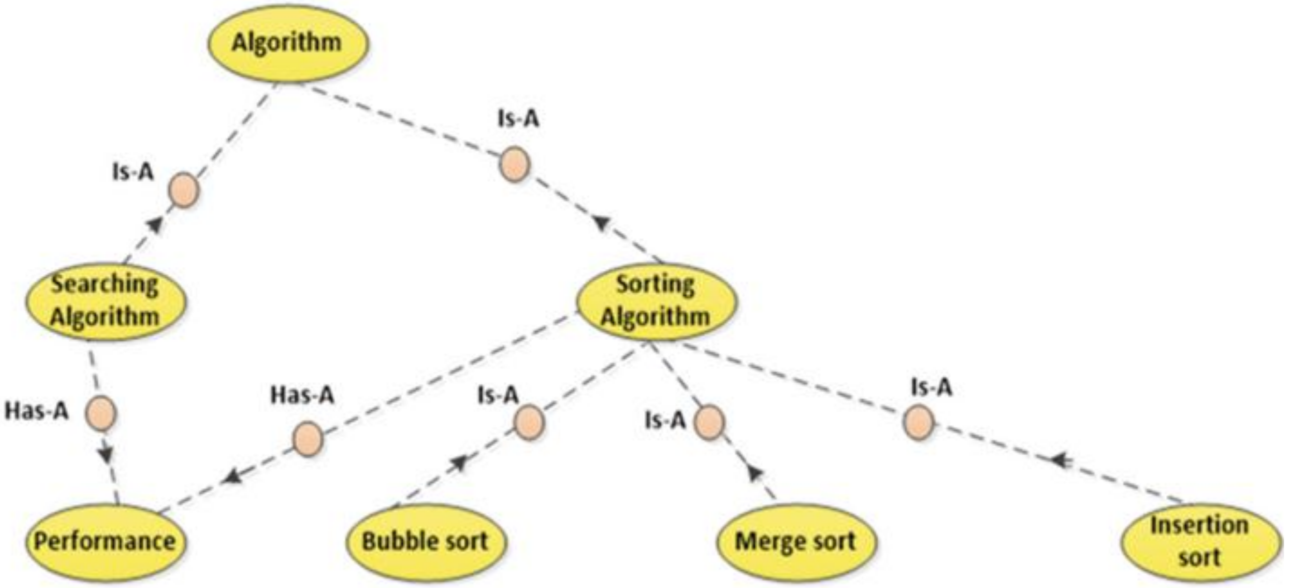
Knowledge Graphs and Ontologies are based on RDF

- RDF, a standard model for data interchange on the Web, uses URIs to name things and the relationship between things, which are referred to as triples:

(1) Subject – (2) Predicate – (3) Object



- **Ontology as Foundation Layer for KG**
  - Ontology: extract taxonomic relations and attributes, plus some semantic relations.
  - Knowledge Graph focuses on extracting relationships in all forms with the same priority.



(**Domain**, Data structure)	// domain
(**Class**, Algorithm)	// c_1
(**SubClass**, Sorting algorithm, Algorithm)	// (r_1, c_2, c_1)
(**SubClass**, Searching algorithm, Algorithm)	// (r_1, c_3, c_1)
(**Has/Property**, Performance, Sorting algorithm)	// (r_2, c_4, c_2)
(**Has/Property**, Performance, Searching algorithm)	// (r_2, c_5, c_3)
(**SubClass**, Bubble sort, Sorting algorithm)	// (r_1, c_6, c_2)
(**SubClass**, Merge sort, Sorting algorithm)	// (r_1, c_7, c_2)
(**SubClass**, Insertion sort, Sorting algorithm)	// (r_1, c_8, c_2)

c_i: concept r_i: relation

- **An example: From Ontologies to Knowledge Graphs**
  - We have three objects: books, authors, and publishers:

Books

Title	Author	Publisher	Year Published	Followed By
To Kill a Mockingbird	Harper Lee	J. B. Lippincott Company	1960	Go Set a Watchman
Go Set a Watchman	Harper Lee	HarperCollins, LLC; Heinemann	2015	
The Picture of Dorian Gray	Oscar Wilde	J. B. Lippincott & Co.	1890	
2001: A Space Odyssey	Arthur C. Clarke	New American Library, Hutchinson	1968	

Publishers

Name	City	Country
J. B. Lippincott & Company	Philadelphia	United States
HarperCollins, LLC	New York City	United States
Heinemann	Portsmouth	United States
New American Library	New York City	United States
Hutchinson	London	United Kingdom

Authors

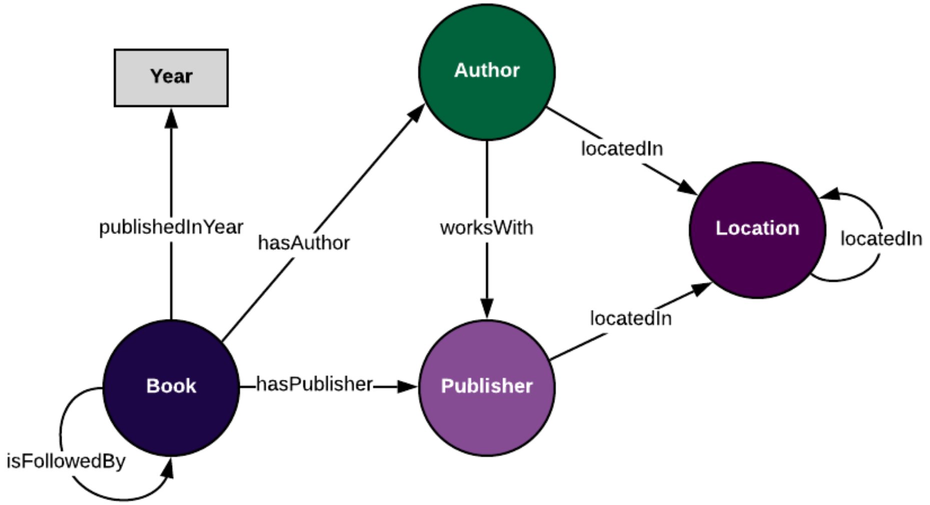
Name	Country of Birth
Harper Lee	United States
Oscar Wilde	Ireland
Arthur C. Clarke	United Kingdom

We define the properties:

- Book → has author → Author
- Book → has publisher→ Publisher
- Book → published on → Publication date
- Book → is followed by → Book
- Author → works with → Publisher
- Publisher → located in → Location
- Location → located in → Location

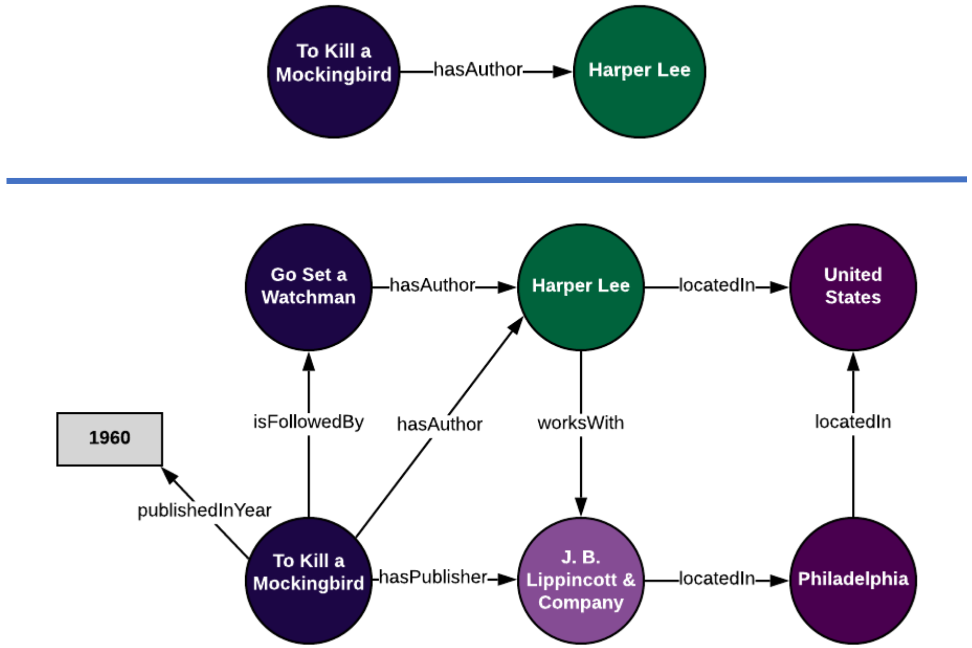
➢ **From Ontologies to Knowledge Graphs: An example**

Ontology



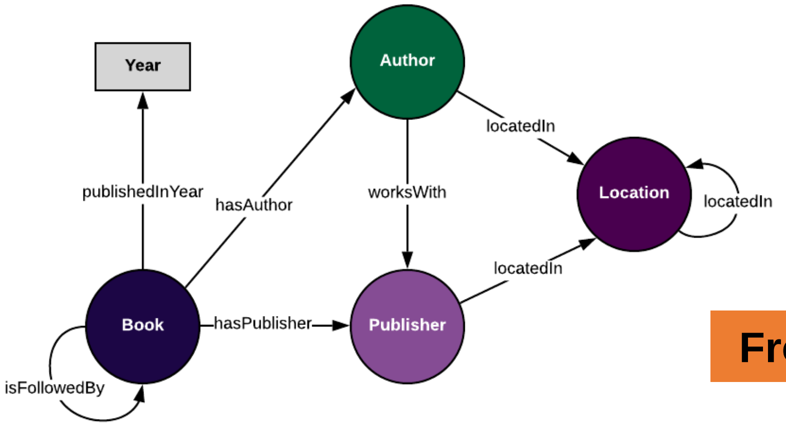
Knowledge graph

Using ontology as a framework, we can add in real data about individual books, authors, publishers, and locations to create a **knowledge graph**

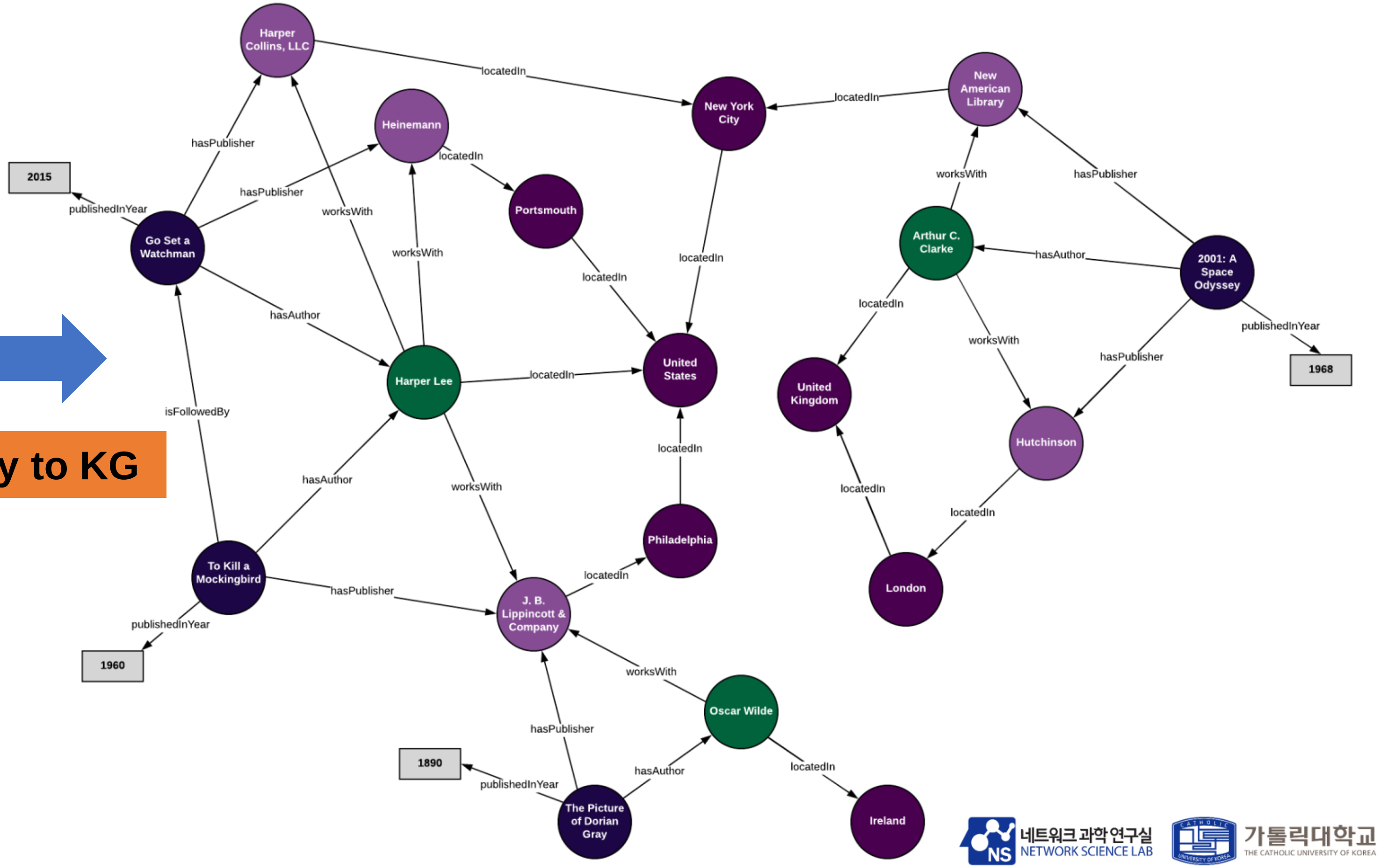


- **Full knowledge graph representation: An example**

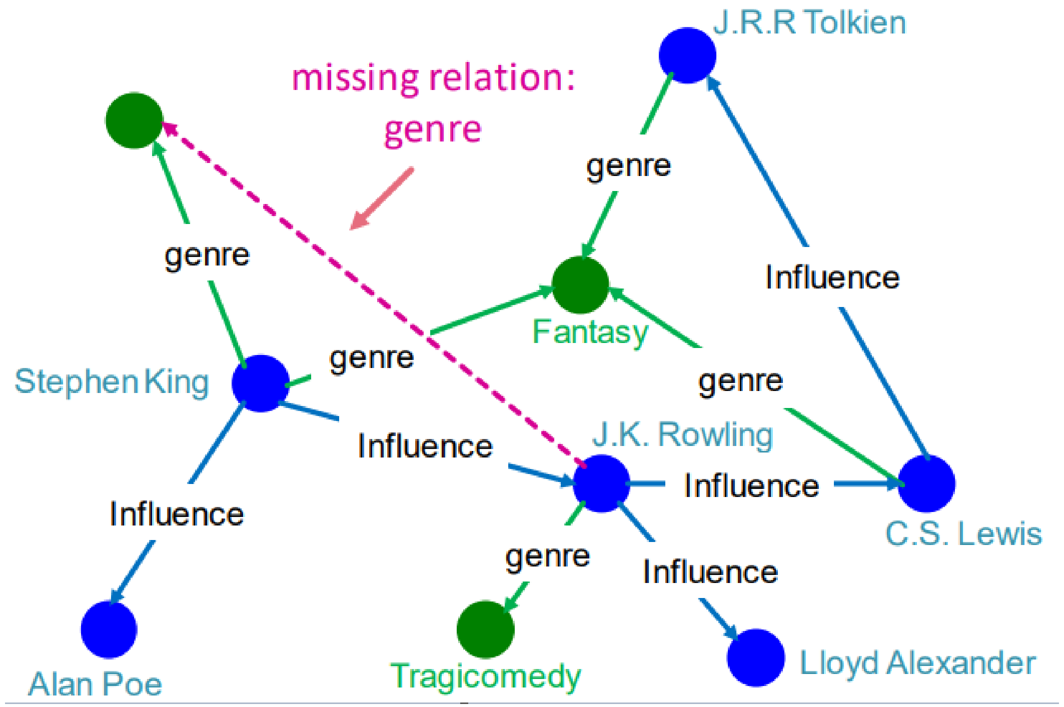
Adding in real data about individual books, authors, publishers, and locations to create a complete KG.



From Ontology to KG

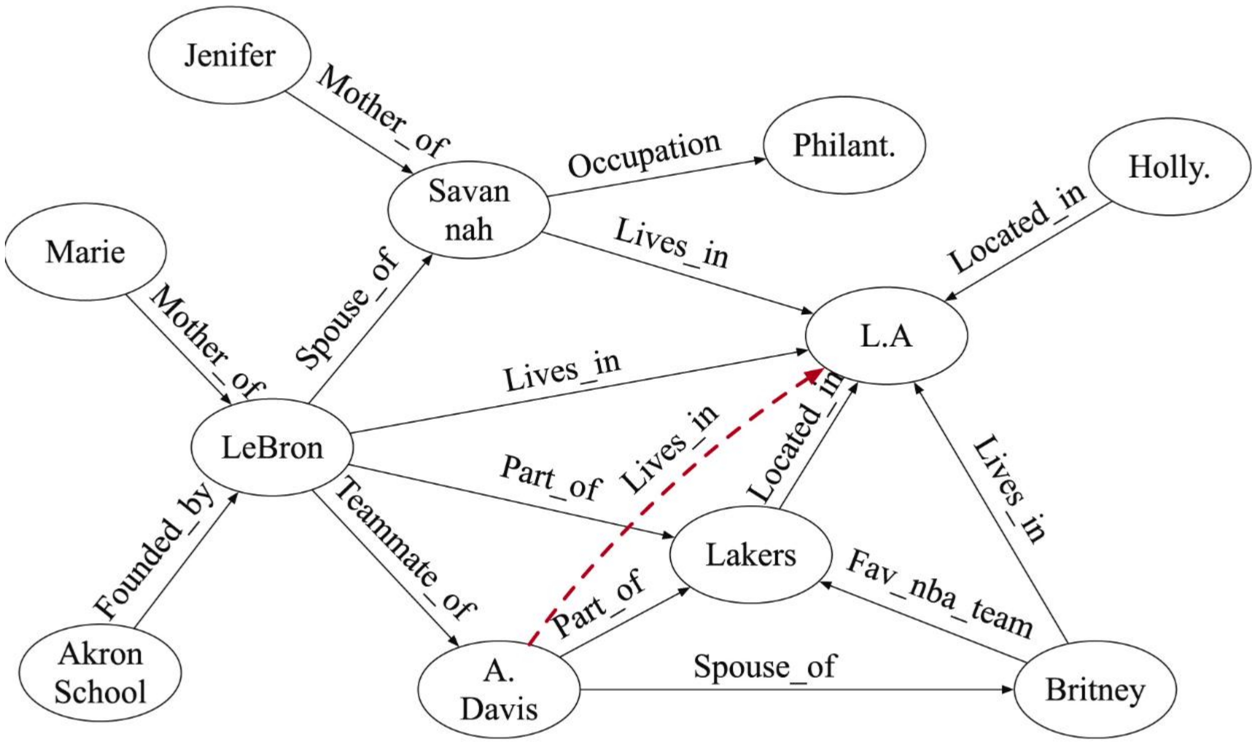


- **Knowledge Graph completion task**
  - Given an enormous KG, can we complete the KG?
  - For a given (head, relation), we predict missing tails

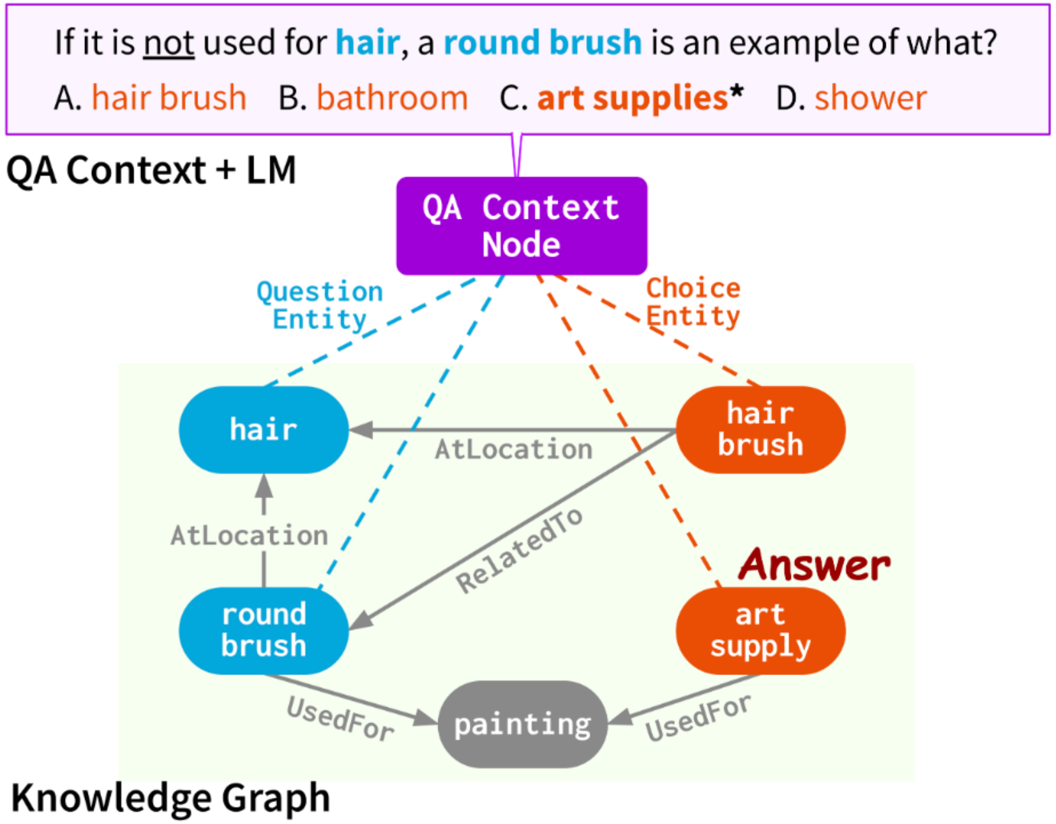


Example task:
predict the tail "Science Fiction" for ("J.K. Rowling", "genre")

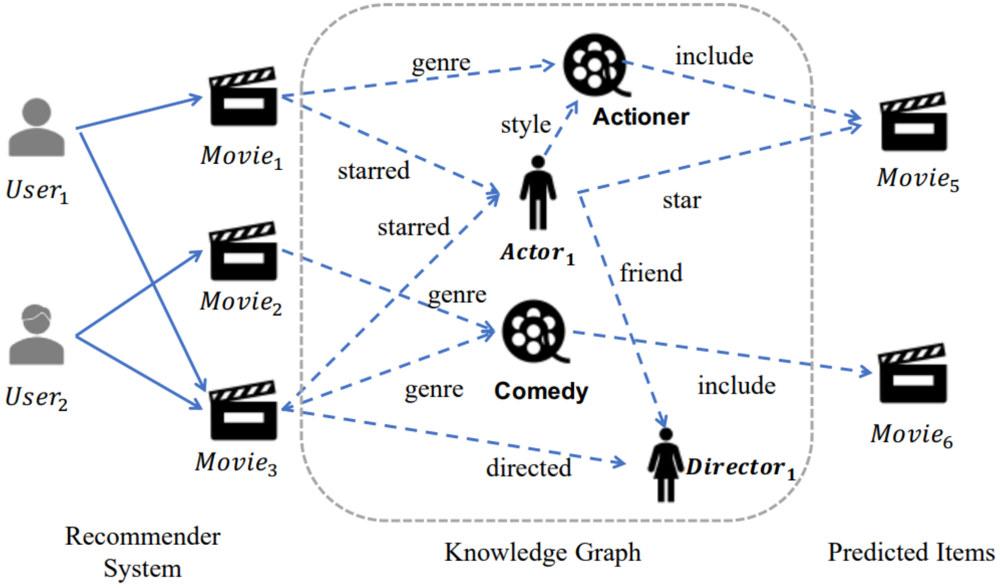
- **Semantic Information**
  - Knowledge graph completion:
  - Query relations:
    - Lives_in
    - Head entity: A.Davis
  - Reasoning result:
    - L.A
- **KG question answering:**
  - Questions:
    - Where do the spouses of teammates of Lakers usually live?
  - Reasoning result:
    - L.A

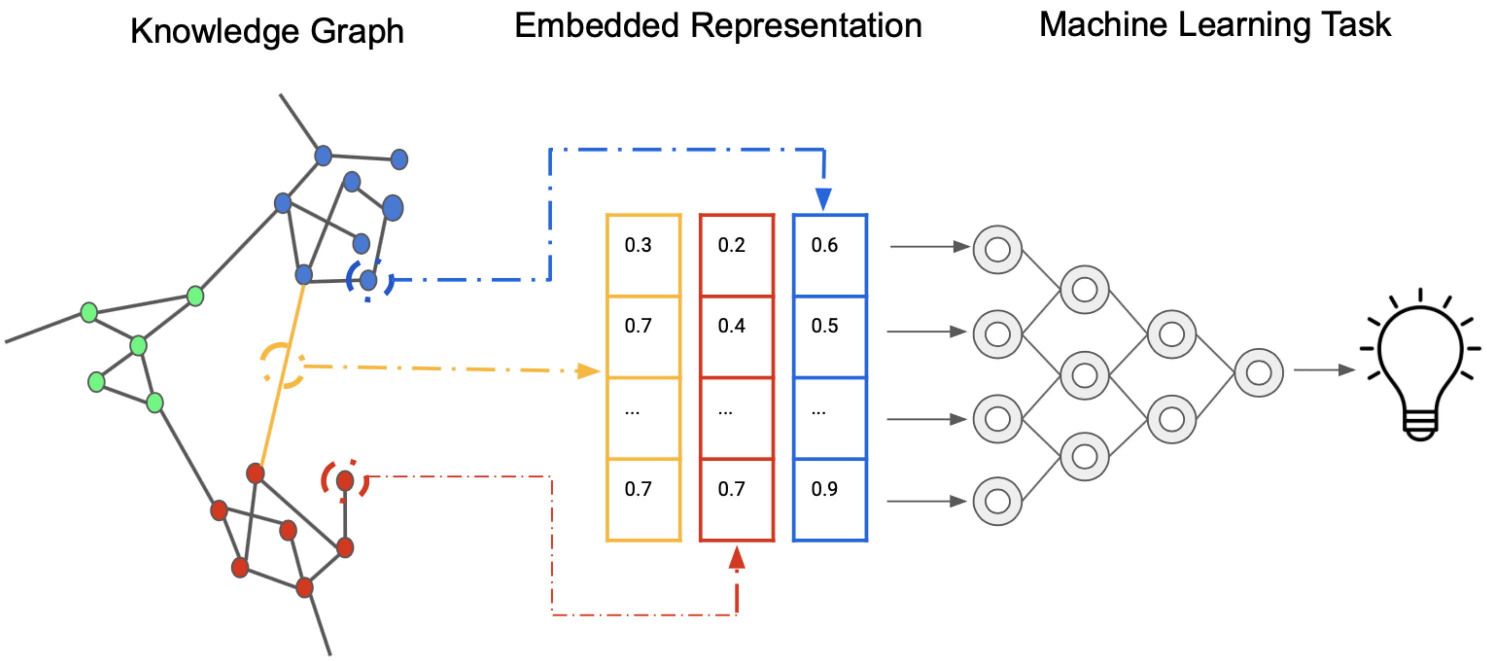


- **Question Answering over Knowledge Graphs**
  - View the QA context as a node (Purple node) and connect it to each topic entity in the KG (blue and red nodes).
  - Each node is associated with one of 4 types:
    - Purple is the QA context node
    - Blue is an entity in the question
    - Orange is an entity in the answer choices
    - Gray is any other entity.
  - The representation is initialized as the LM representation of the QA context or entity name.



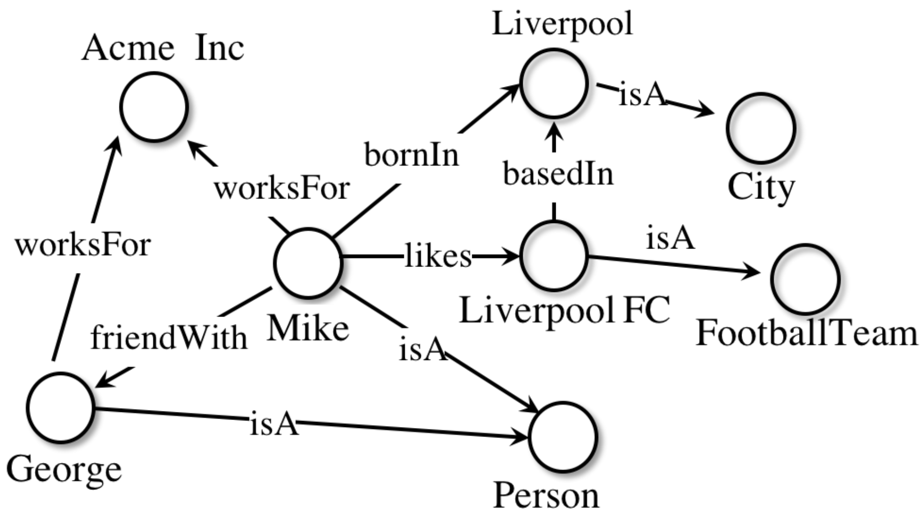
- **Recommender system**
  - KGs have been used in recommender systems in order to overcome the problem of user-item interactions sparsity and the cold start problem.
  - The vector representation of the entities and relations can be used for different machine learning applications.

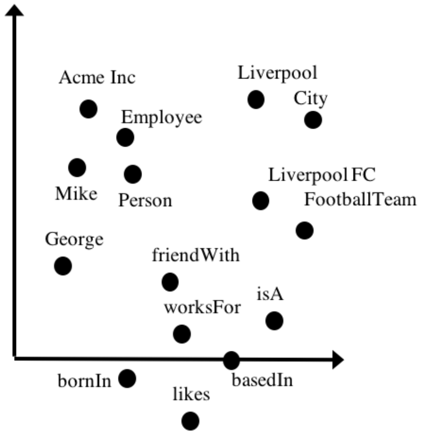


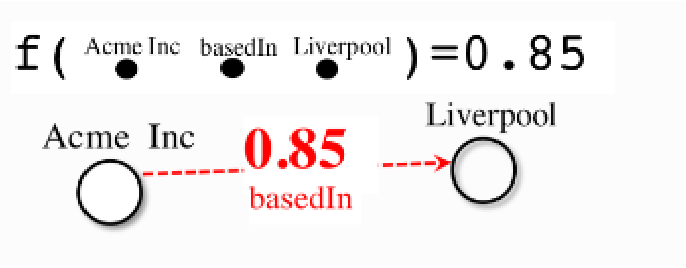


Knowledge Graph Representation Learning

- **Mapping from Graph domain to Space domain**
  - Use Graph embeddings for a latent semantic representation of Knowledge Graphs
  - Combining latent semantic representations of different (symbolic) representations

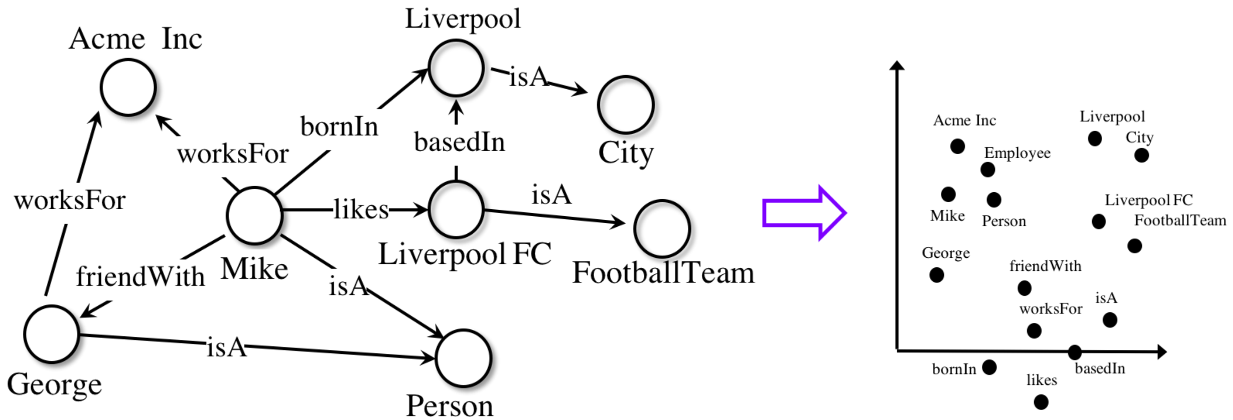






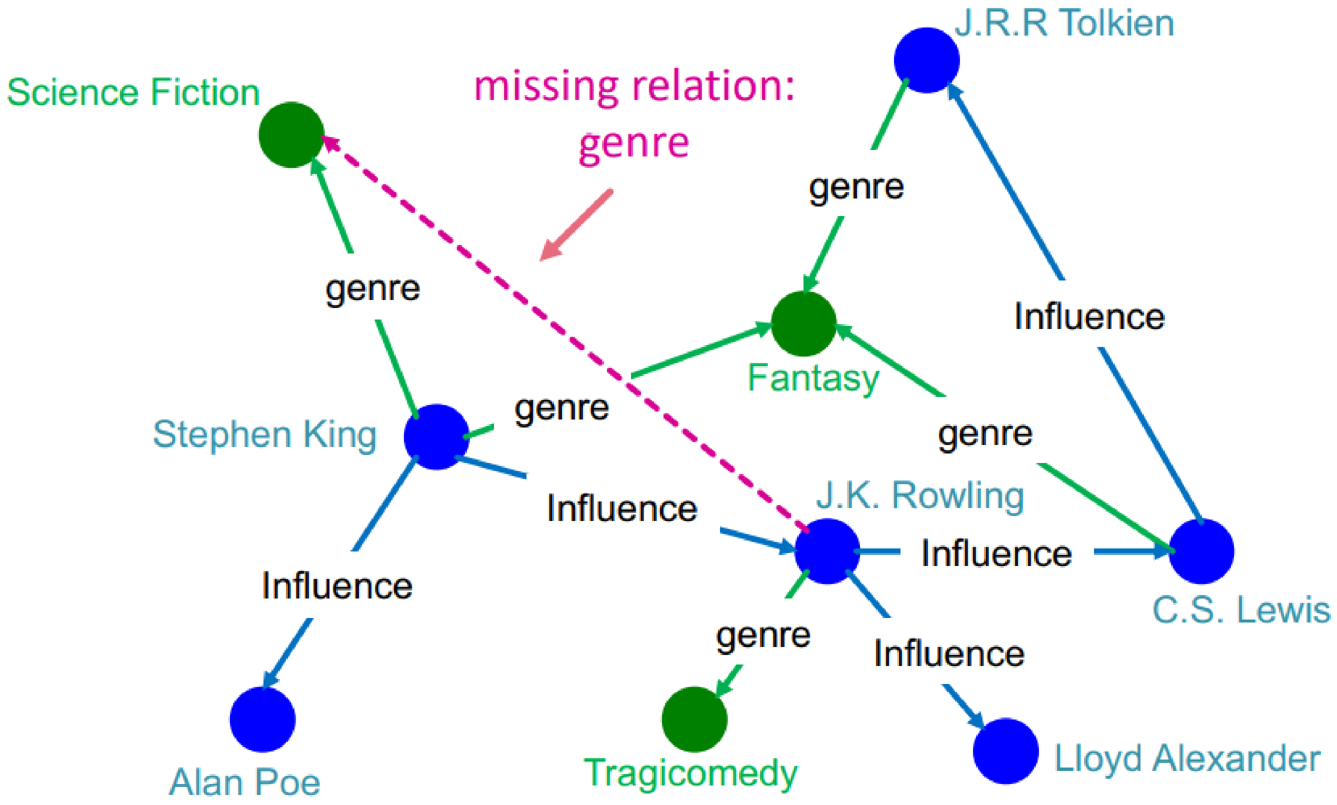
A scoring function is used to measure the plausibility of a triple

- **Why do we need vector embeddings?**
  - Embeddings make it easier to do machine learning on large inputs like sparse word vectors.
- **Where do the embeddings come from?**
  - arned from the knowledge base itself (e.g. KG completion)
  - Learned from text (e.g. word embeddings)
- **What is the underlying principle?**
  - Similarity-based reasoning is highly heuristic.
  - No strong reason to believe that something is true just because it is true for a similar predicate or individual.

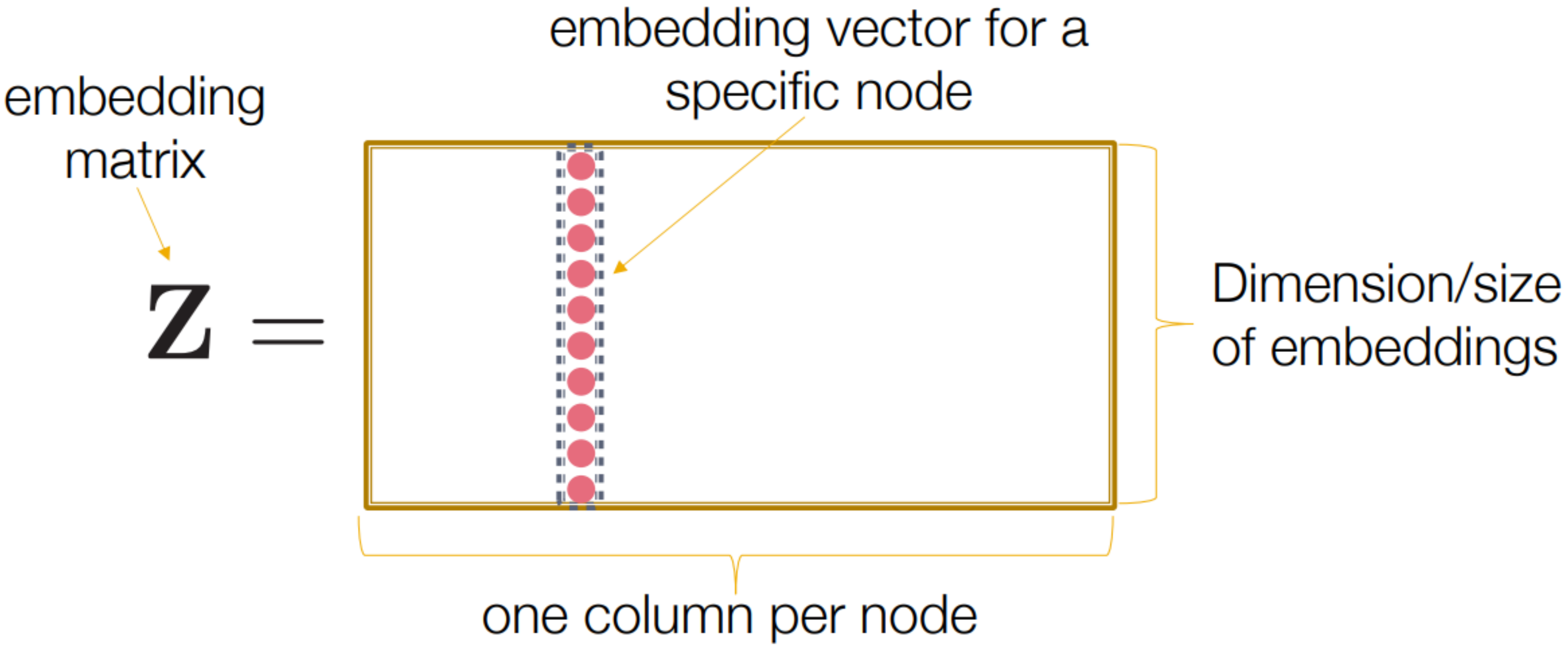


Knowledge Graph completion

- 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”)



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



Knowledge Graph 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 R^d
  - Associate entities and relations with shallow embeddings

  - Given a true triple *(h, r, t)*, the goal is that the embedding of (h, r) should be close to the embedding of t.
- **Main questions:**
  - How to embed (h, r) ?
  - How to define the similarity between them?

Translation-based Embedding Approaches

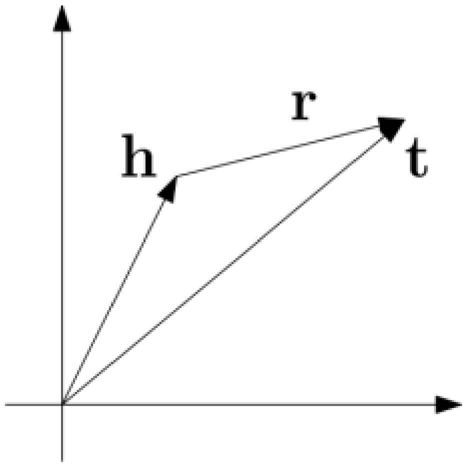
- Focused on embedding monolingual triples (h, r, t)
  - Exploit distance-based scoring functions
  - Measure the plausibility of a fact as the distance between the two entities

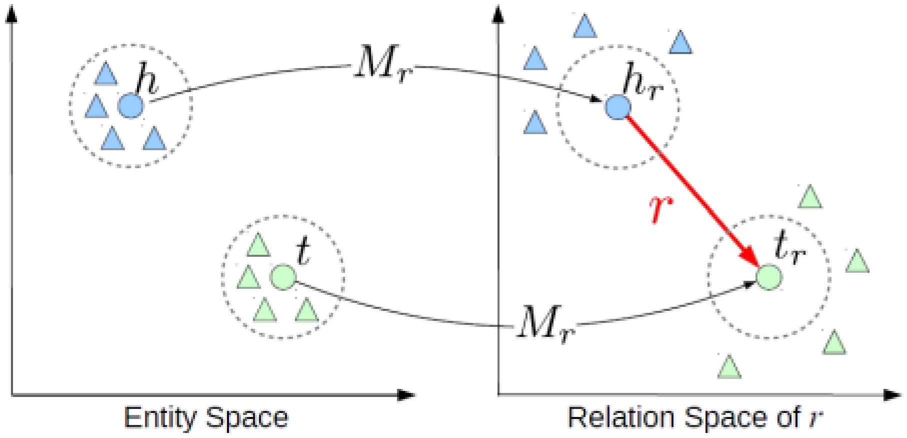
TransE: h+r ≈ t

Later approaches:

- TransH [Wang et al. 2014]
- TransR [Lin et al. 2015]
- TransD [Ji et al. 2015]
- HolE [Nickle et al. 2016]
- ComplEx [Trouillon et al. 2016]

Embedding of monolingual knowledge seems to be well-addressed.

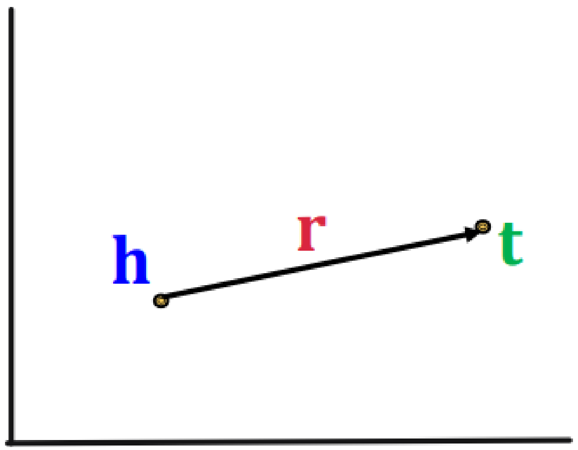


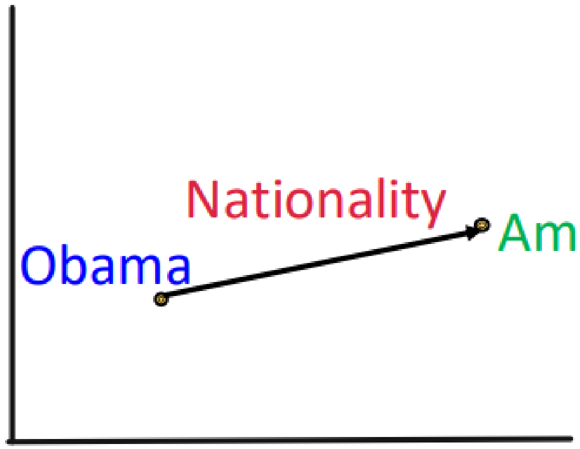


Translation-based Embedding Approaches: TransE

- For a triple (h, r, t):
  - $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ if the given fact is true
  - else $\mathbf{h} + \mathbf{r} \neq t$

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





Bordes, Antoine, et al. (2013) Translating embeddings for modeling multi-relational data. Advances in neural information processing systems.

Relation Patterns

- **Symmetric** (Antisymmetric) Relations:

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

  - Example:
    - Symmetric: Family, Roommate
    - Antisymmetric: Hypernym

- **Inverse** Relations:
  - Symmetric (Antisymmetric) Relations: $r_2(h,t) \Rightarrow r_1(t,h)$
  - Example : (Advisor, Advisee)

- **1-to-N** relations:

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

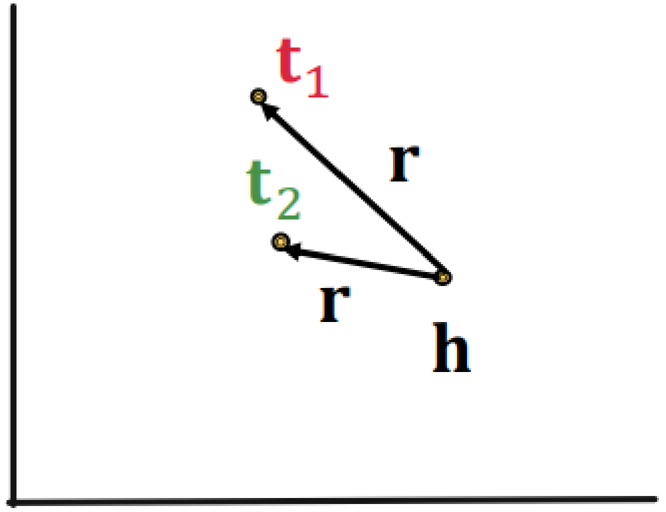
  - Example: *r* is "StudentsOf"

Translation-based Embedding Approaches: TransE

- TransE Limitation: 1-to-N relations
  - 1-to-N Relations: (h, r, t_1) and (h, r, t_2) both exist in the knowledge graph.
  - TransE cannot model 1-to-N relations: t_1 and 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!



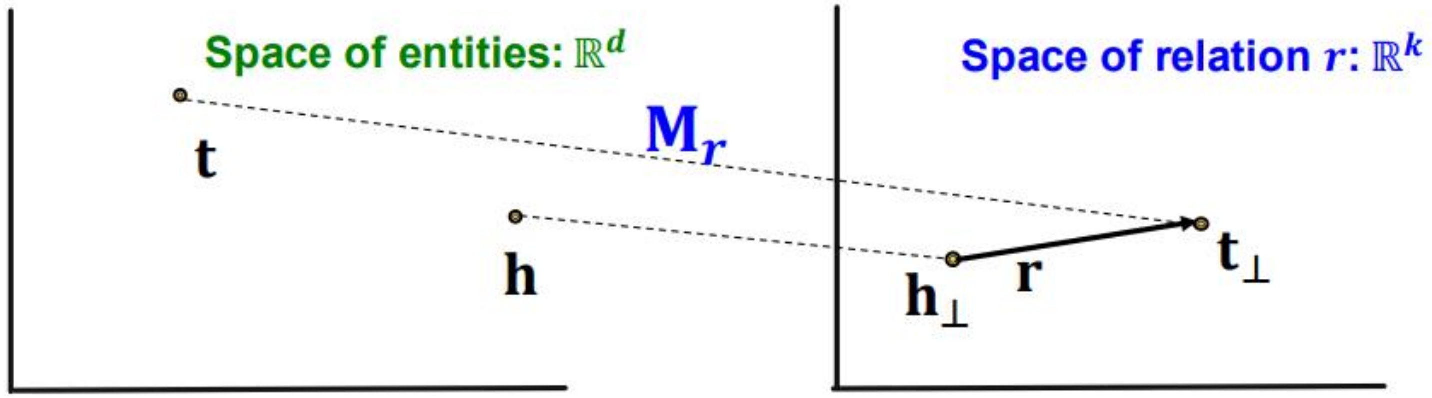
Bordes, Antoine, et al. (2013) Translating embeddings for modeling multi-relational data. Advances in neural information processing systems.

Translation-based Embedding Approaches: TransR

- TransE models the 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 $M_r \in \mathbb{R}^{k \times d}$ as the projection matrix.



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

Translation-based Embedding Approaches: TransR

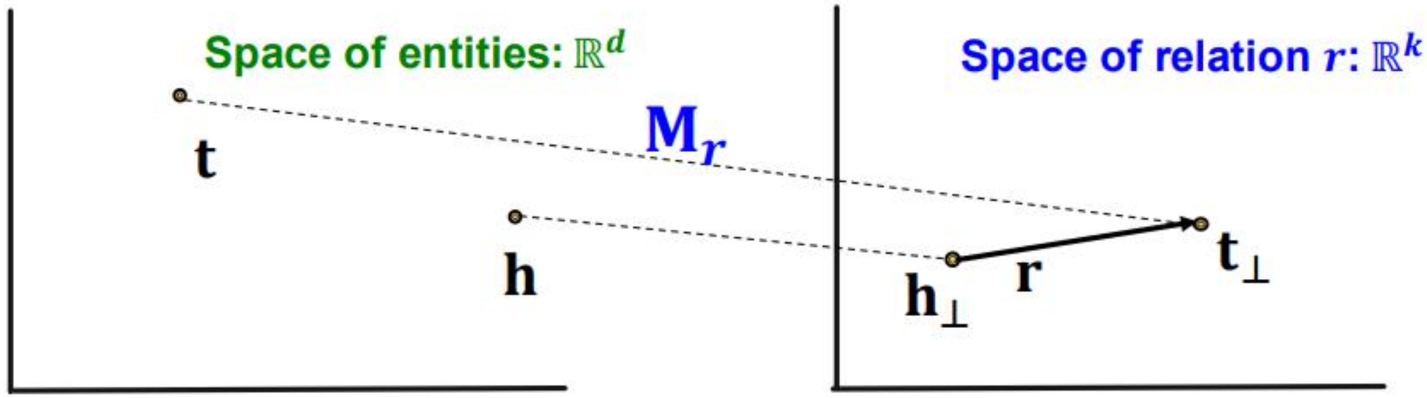
- **TransR**: model entities as vectors in the entity space $\mathbb{R}^d$ and model each relation as vector in relation space $\mathbf{r} \in \mathbb{R}^k$ with $M_r \in \mathbb{R}^{k\times d}$ as the projection matrix:

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

- Score function:

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

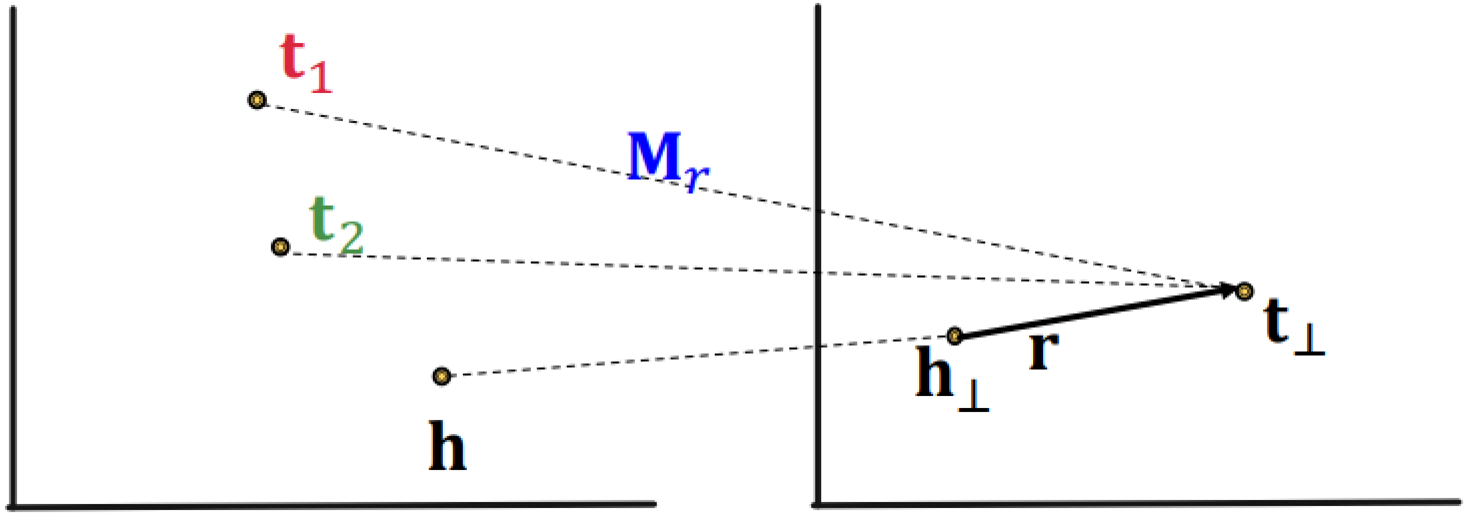
Use M_r to project from entity space $\mathbb{R}^d$ to relation space $\mathbb{R}^k$



- TransR: 1-to-N relations in TransR
  - 1-to-N Relations:
    - Example: If (h, r, t_1) and (h, r, t_2) exist in the knowledge graph.
  - TransR can model 1-to-N relations
    - We can learn $\mathbf{M}_r$ so that:

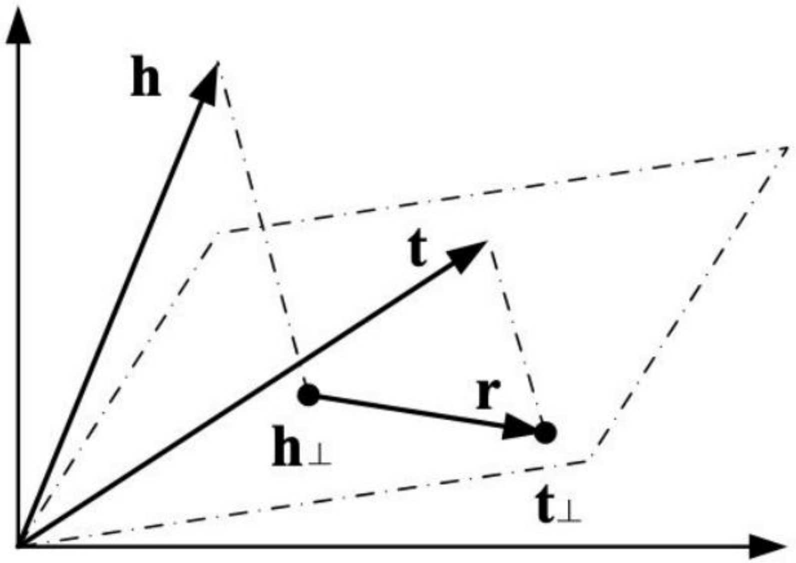
$$\mathbf{t}_\perp = \mathbf{M}_r \mathbf{t}_1 = \mathbf{M}_r \mathbf{t}_2$$

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



Translation-based Embedding Approaches: TransH

- From Original space to Hyperplane
- TransH enables different toles of an entity in different relations
- Entities h and t are projected into specific hyperplane of relation r
- Then predict new links based on translation on hyperplane

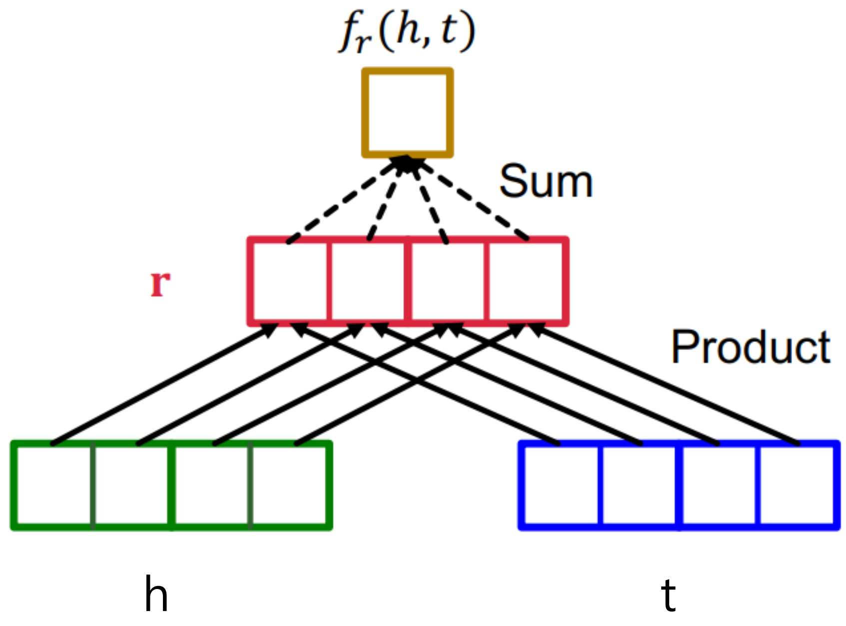


Wang, Zhen, et al. Knowledge graph embedding by translating on hyperplanes (AAAI 2014)

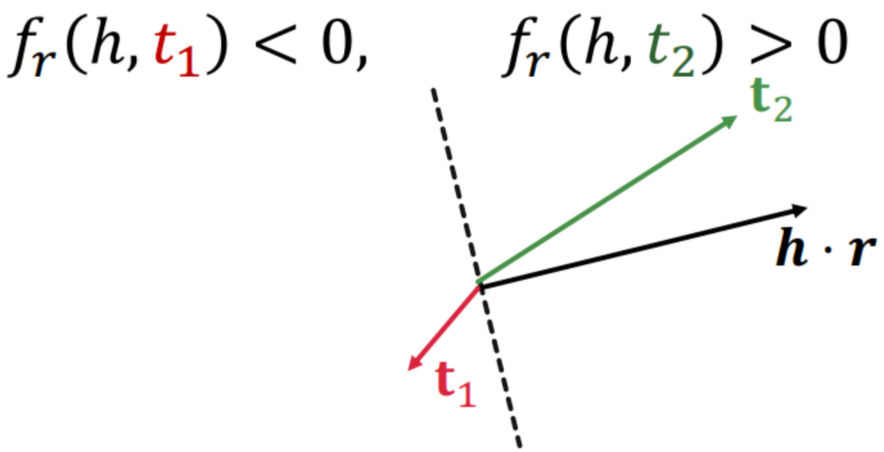
Translation-based Embedding Approaches: DistMult

- So far: The scoring function $f_r(h,t)$ is negative of L1 / L2 distance in TransE and TransR
- Another line of KG embeddings adopt bilinear modelling
- **DistMult**: Entities and relations using vectors in R^k
- Score function:

$$f_r(h,t) =< \mathbf{h}, \mathbf{r}, \mathbf{t} > = \sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i \qquad \mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$$



- **DistMult**: Entities and relations using vectors in R^k
- Intuition of the score function: Can be viewed as a cosine similarity between h · r and t
  - where h · r is defined as $\sum_i \boldsymbol{h_i} \cdot \boldsymbol{r_i}$
  - Example:



네트워크 과학 연구실
NETWORK SCIENCE LAB

가톨릭대학교
THE CATHOLIC UNIVERSITY OF KOREA