# GNN - III

## - Heterogeneous Graphs & RGCN

# Heterogeneous Graphs & RGCN

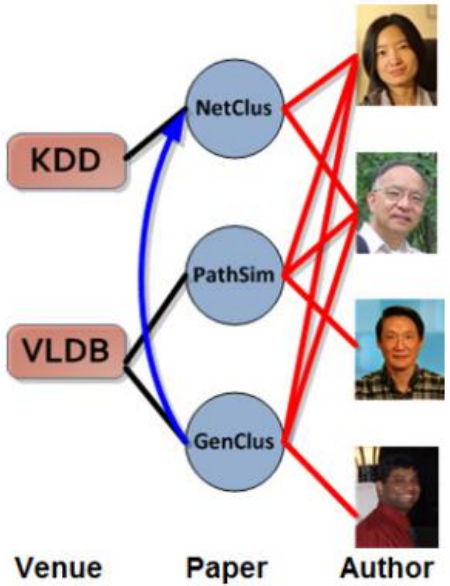
- Heterogeneous Graphs
- Relational Graph Convolutional Networks
- Entity classification
- Link prediction

# Heterogeneous Graphs

- Heterogeneous graphs are graphs that contain different types of nodes and edges which have different types of attributes that are designed to capture the characteristics of each node and edge type.
- Within the context of graph neural networks, certain node and edge types might need to be modeled with different number of dimensions.

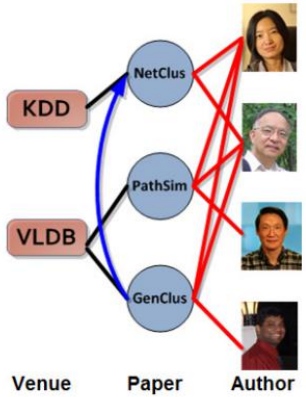
# Examples of heterographs

- Citation graph: The Association for Computing Machinery publishes an ACM dataset that contains two million papers, their authors, publication venues, and the other papers that were cited.



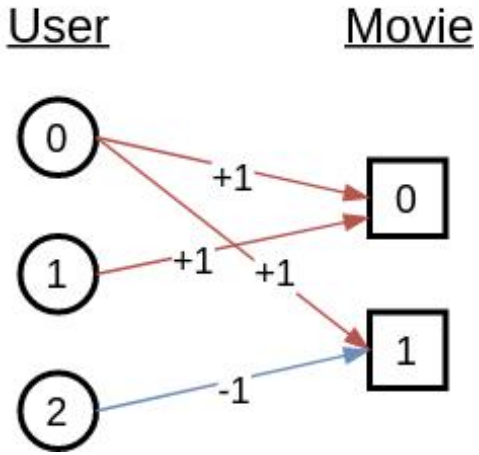
# Examples of heterographs

- It has three types of entities that correspond to **papers**, **authors**, and **publication venues**.
- It contains three types of edges :
  - Authors with papers corresponding to **written-by** relationships
  - Papers with venues corresponding to **published-in** relationships
  - Papers with other papers corresponding to **cited-by** relationships



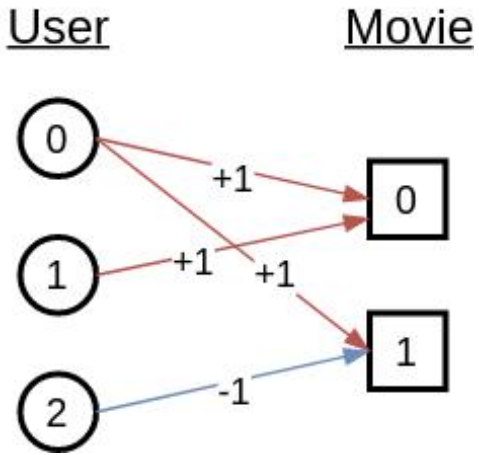
# Examples of heterographs

- Recommender systems: The datasets used in recommender systems often contain interactions between users and items. For example, the data could include the ratings that users have provided to movies.



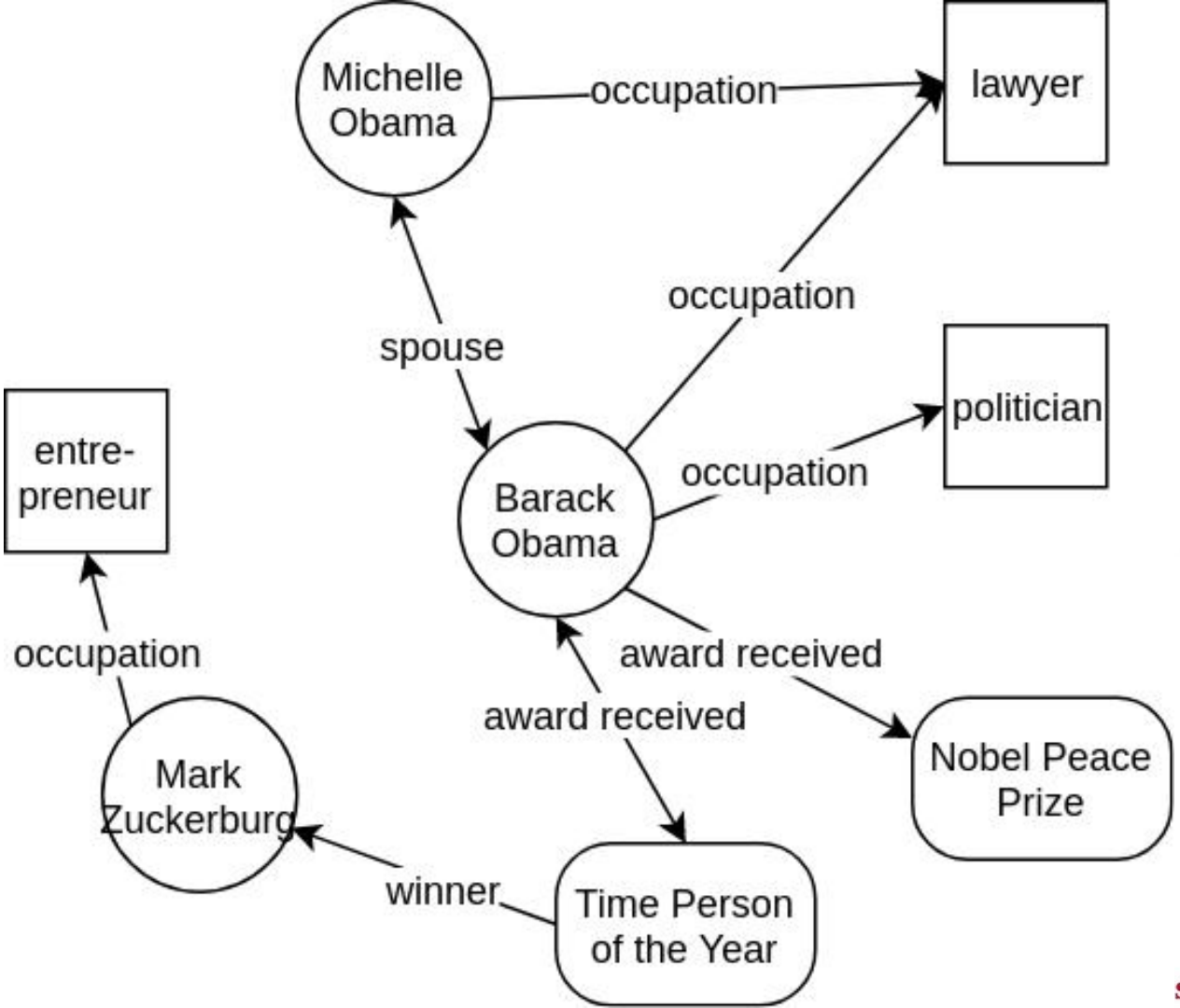
# Examples of heterographs

- The nodes in these heterographs will have two types, users and movies.
- The edges will correspond to the user-movie interactions. Furthermore, if an interaction is marked with a rating, then each rating value could correspond to a different edge type.



# Examples of heterographs

- Knowledge graph: Knowledge graphs are inherently heterogenous. For example, in Wikidata, Barack Obama (item Q76) is an instance of a human, which could be viewed as the entity class, whose spouse (item P26) is Michelle Obama (item Q13133) and occupation (item P106) is politician (item Q82955).



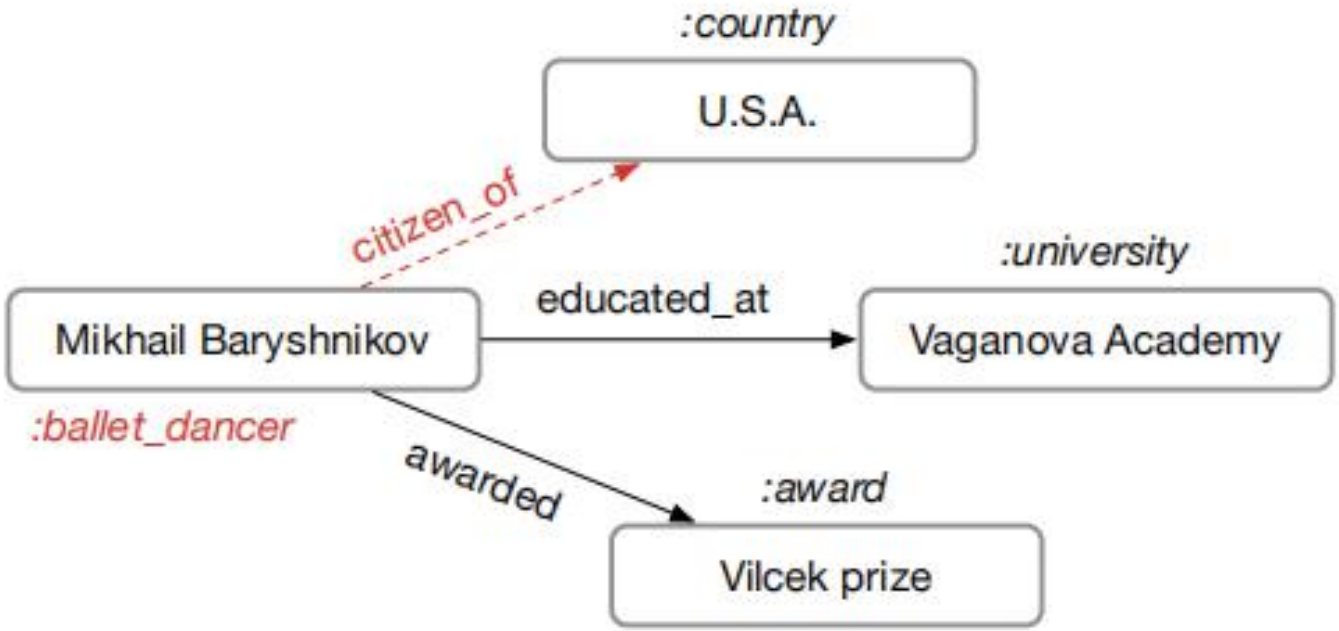



Figure 1: A knowledge base fragment: The nodes are entities, the edges are relations labeled with their types, the nodes are labeled with entity types (e.g., *university*). The edge and the node label shown in red are the missing information to be inferred.

# Manipulating heterograph

- See the code...

# Learning tasks with heterographs

- **Node classification/regression** to predict the class of each node or estimate a value associated with it.
- **Link prediction** to predict if there is an edge of a certain type between a pair of nodes, or predict which other nodes a particular node is connected with (and optionally the edge types of such connections).
- **Graph classification/regression** to assign an entire heterograph into one of the target classes or to estimate a numerical value associated with it.

# semi-supervised node classification

- Our goal is to predict the publishing conference of a paper using the ACM academic graph we just created. To further simplify the task, we only focus on papers published in three conferences: KDD, ICML, and VLDB. All the other papers are not labeled, making it a semi-supervised setting.

We use Relational-GCN to learn the representation of nodes in the graph. Its message-passing equation is as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_r(i)} W_r^{(l)} h_j^{(l)}\right)$$

Breaking down the equation, you see that there are two parts in the computation.

1. Message computation and aggregation within each relation $r$
2. Reduction that merges the results from multiple relationships

Following this intuition, perform message passing on a heterograph in two steps.

1. Per-edge-type message passing
2. Type wise reduction

# Relational-GCN on heterograph

- See the code...

# R-GCN

- The relational graph convolutional network (R-GCN) is one effort to generalize GCN to handle different relationships between entities in a knowledge base.

# R-GCN

- A knowledge graph is made up of a collection of triples in the form subject, relation, object. Edges thus encode important information and have their own embeddings to be learned. Furthermore, there may exist multiple edges among any given pair.

# A brief introduction to R-GCN

In statistical relational learning (SRL), there are two fundamental tasks:

- Entity classification - Where you assign types and categorical properties to entities.
- Link prediction - Where you recover missing triples.

In both cases, missing information is expected to be recovered from the neighborhood structure of the graph.

# A brief introduction to R-GCN

R-GCN solves these two problems using a common graph convolutional network. It's extended with multi-edge encoding to compute embedding of the entities, but with different downstream processing.

- Entity classification is done by attaching a softmax classifier at the final embedding of an entity (node). Training is through loss of standard cross-entropy.
- Link prediction is done by reconstructing an edge with an autoencoder architecture, using a parameterized score function. Training uses negative sampling.

# Key ideas of R-GCN

Recall that in GCN, the hidden representation for each node $i$ at $(l+1)^{th}$ layer is computed by:

$$h_i^{l+1} = \sigma\left(\sum_{j \in N_i} \frac{1}{c_i} W^{(l)} h_j^{(l)}\right) \qquad (1)$$

where $c_i$ is a normalization constant.

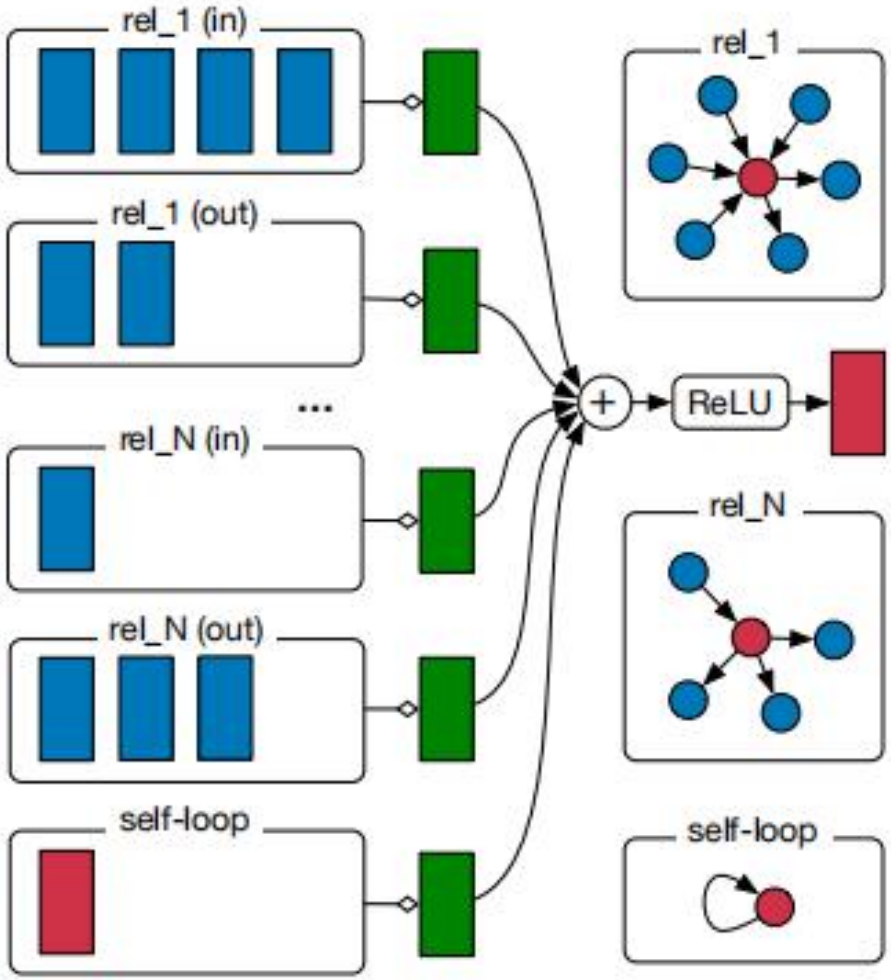
The key difference between R-GCN and GCN is that in R-GCN, edges can represent different relations. In GCN, weight $W^{(l)}$ in equation $(1)$ is shared by all edges in layer $l$. In contrast, in R-GCN, different edge types use different weights and only edges of the same relation type $r$ are associated with the same projection weight $W_r^{(l)}$.

# Key ideas of R-GCN

So the hidden representation of entities in $(l+1)^{th}$ layer in R-GCN can be formulated as the following equation:

$$h_i^{l+1} = \sigma\left(W_0^{(l)} h_i^{(l)} + \sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)}\right) \qquad (2)$$

where $N_i^r$ denotes the set of neighbor indices of node $i$ under relation $r \in R$ and $c_{i,r}$ is a normalization constant. In entity classification, the R-GCN paper uses $c_{i,r} = |N_i^r|$.



# Key ideas of R-GCN

The problem of applying the above equation directly is the rapid growth of the number of parameters, especially with highly multi-relational data. In order to reduce model parameter size and prevent overfitting, the original paper proposes to use basis decomposition.

$$W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} V_b^{(l)} \qquad (3)$$

Therefore, the weight $W_r^{(l)}$ is a linear combination of basis transformation $V_b^{(l)}$ with coefficients $a_{rb}^{(l)}$. The number of bases $B$ is much smaller than the number of relations in the knowledge base.

# Key ideas of R-GCN

Another method regularizing the weights of R-GCN-layers: the block-diagonal decomposition, we let each $W^{(l)}_r$ be defined through the **direct sum** over a set of low-dimensional matrices:

$$W_r^{(l)} = \bigoplus_{b=1}^{B} Q_{br}^{(l)}. \tag{4}$$

Thereby, $W_r^{(l)}$ are block-diagonal matrices: $\mathrm{diag}(Q_{1r}^{(l)}, \ldots, Q_{Br}^{(l)})$ with $Q_{br}^{(l)} \in \mathbb{R}^{(d^{(l+1)}/B) \times (d^{(l)}/B)}$.

# Implement R-GCN in DGL

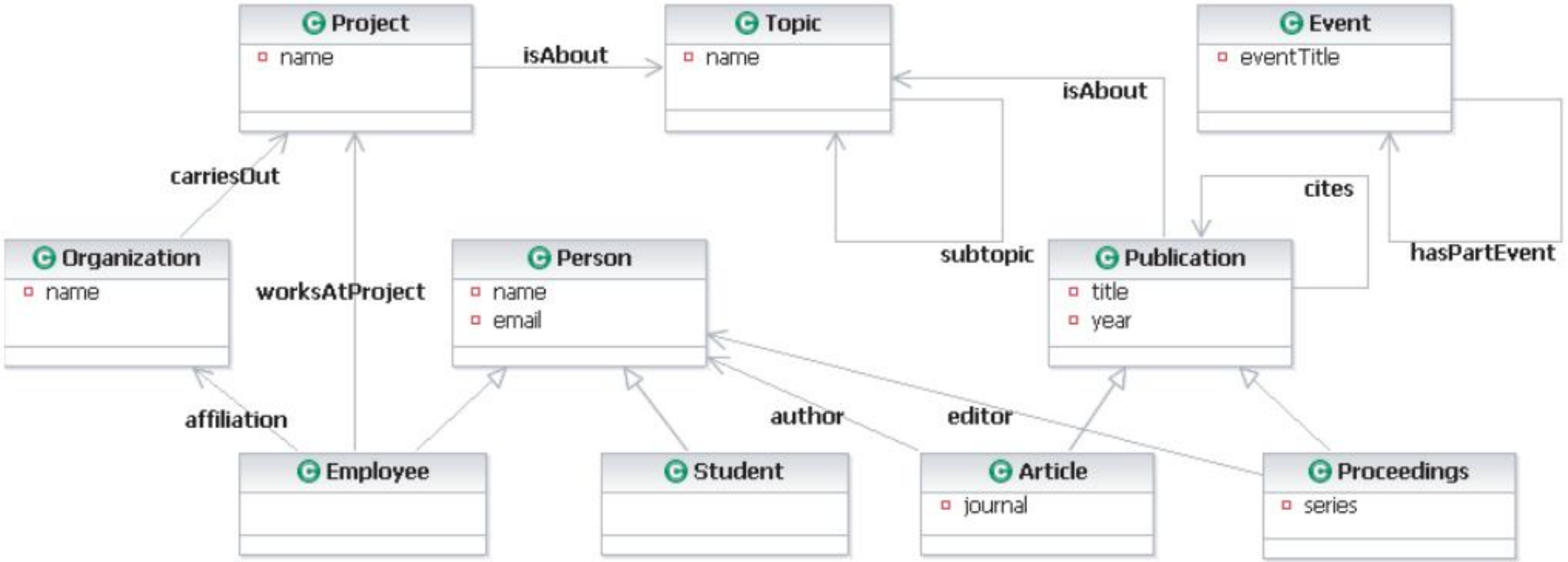
An R-GCN model is composed of several R-GCN layers. The first R-GCN layer also serves as input layer and takes in features (for example, description texts) that are associated with node entity and project to hidden space. In this tutorial, we only use the entity ID as an entity feature.

# R-GCN layers

For each node, an R-GCN layer performs the following steps:

- Compute outgoing message using node representation and weight matrix associated with the edge type (message function)
- Aggregate incoming messages and generate new node representations (reduce and apply function)

# AIFB Dataset



# entity classification

- The entity classification model uses softmax classifiers at each node in the graph. The classifiers take node representations supplied by a relational graph convolutional network (R-GCN) and predict the labels. The model, including R-GCN parameters, is learned by optimizing the cross-entropy loss.

# entity classification

- See the code.

# link prediction

- In the knowledge base setting, representation generated by R-GCN can be used to uncover potential relationships between nodes. In the R-GCN paper, the authors feed the entity representations generated by R-GCN into the DistMult prediction model to predict possible relationships.
- The implementation is similar to that presented here, but with an extra DistMult layer stacked on top of the R-GCN layers.

# link prediction

- The link prediction model can be regarded as an autoencoder consisting of (1) an encoder: an R-GCN producing latent feature representations of entities, and (2) a decoder: a tensor factorization model exploiting these representations to predict labeled edges.