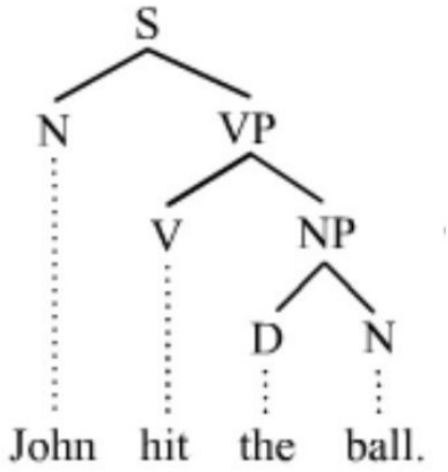
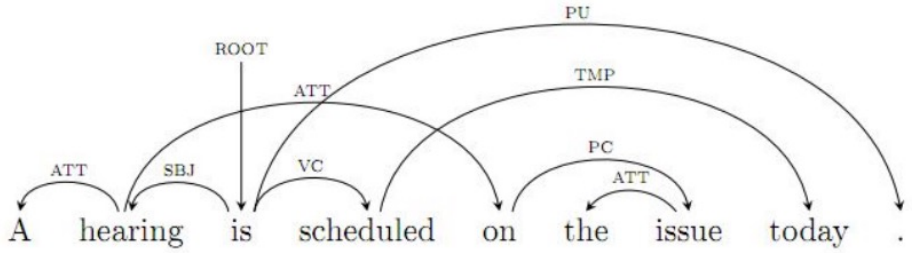
A Tutorial on Graph Neural Networks for Natural Language Processing

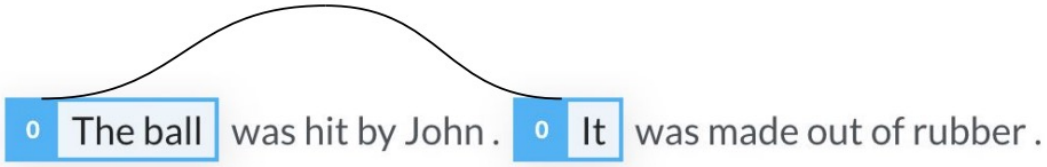
EMNLP 2019 Tutorial

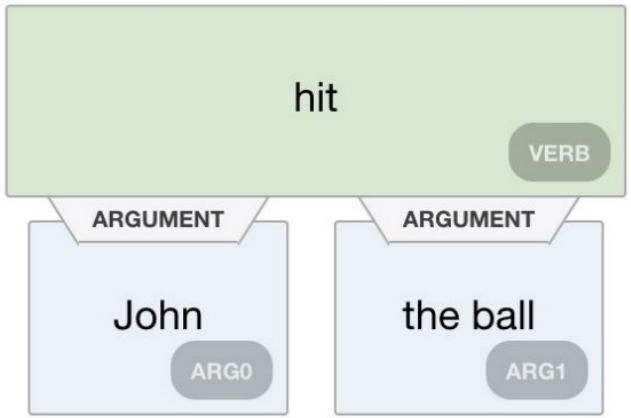
github/svjan5/GNNs-for-NLP

Graphs are everywhere in NLP

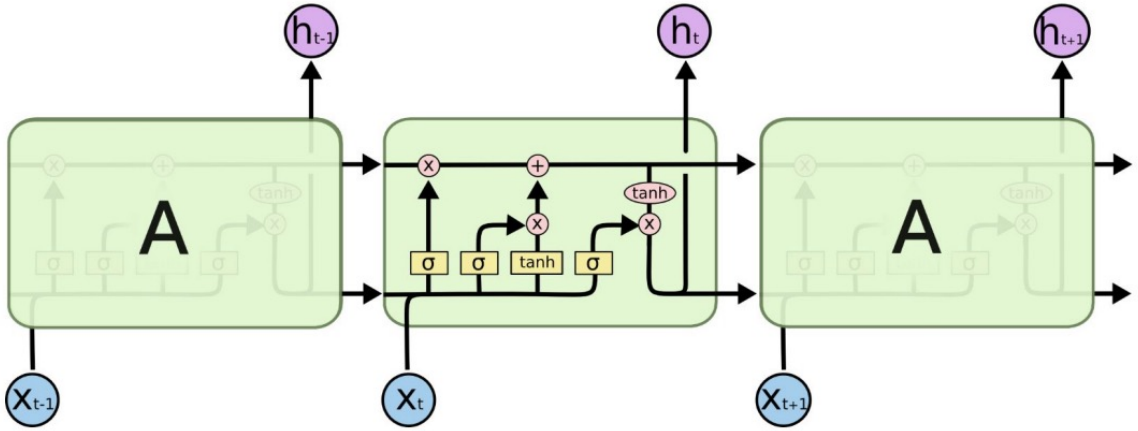




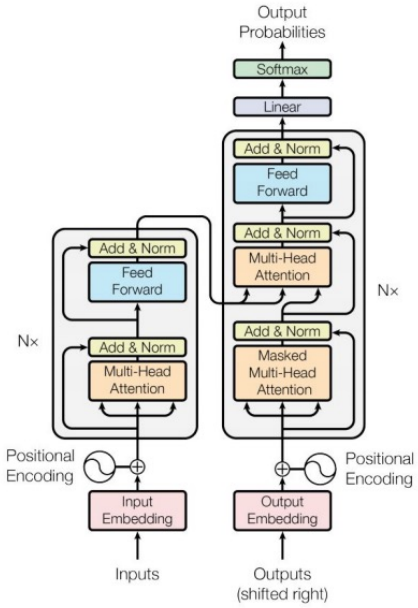




Deep Learning in NLP



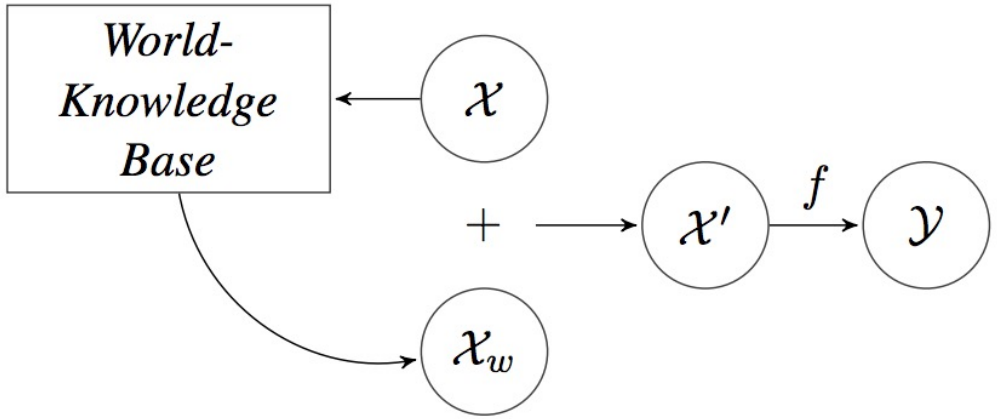
Recurrent Networks



Transformers

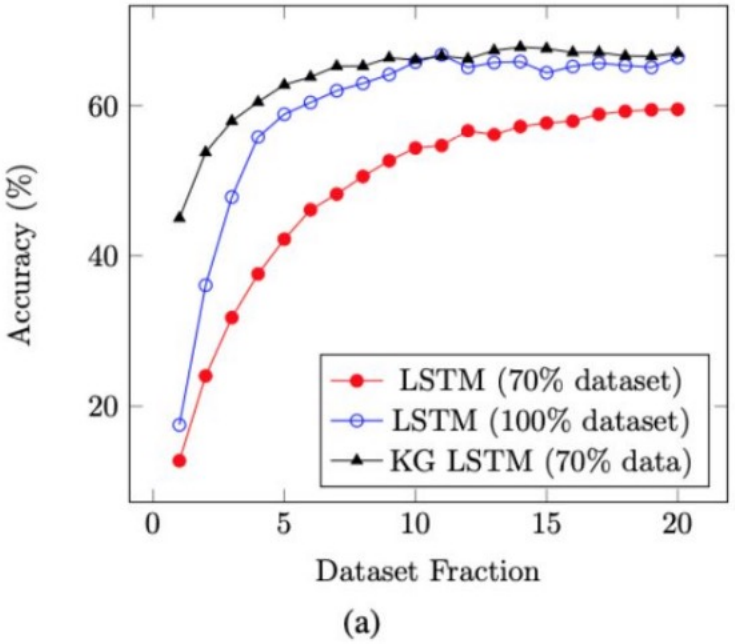
It is not clear how to incorporate graph structures.

Why Deep Learning over Graph for NLP?



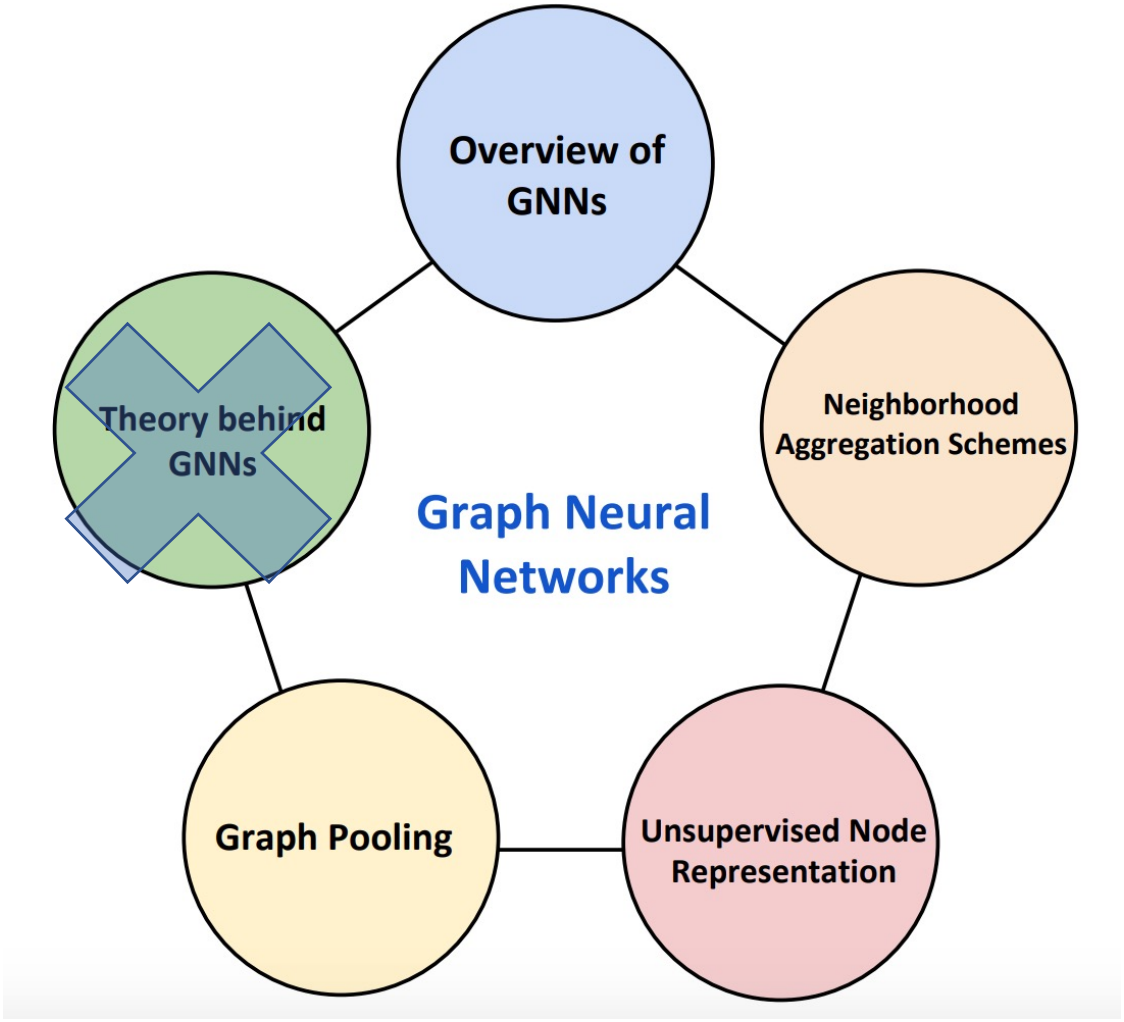
Donald Trump offered his condolences towards the hurricane victims and their families in Texas. => **Politics**

<Donald Trump, president, United States>
<Texas, state, United States>

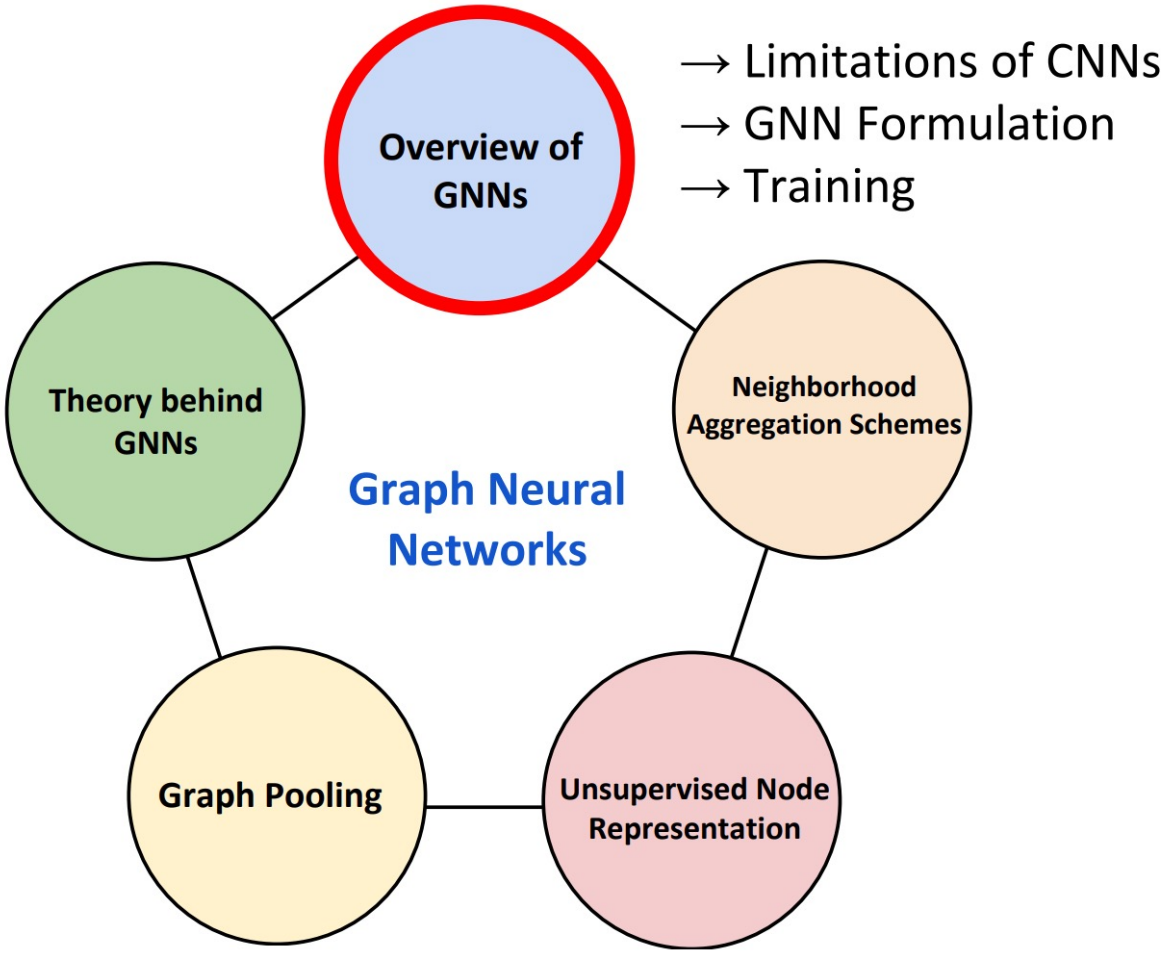


News20

Graph Neural Networks

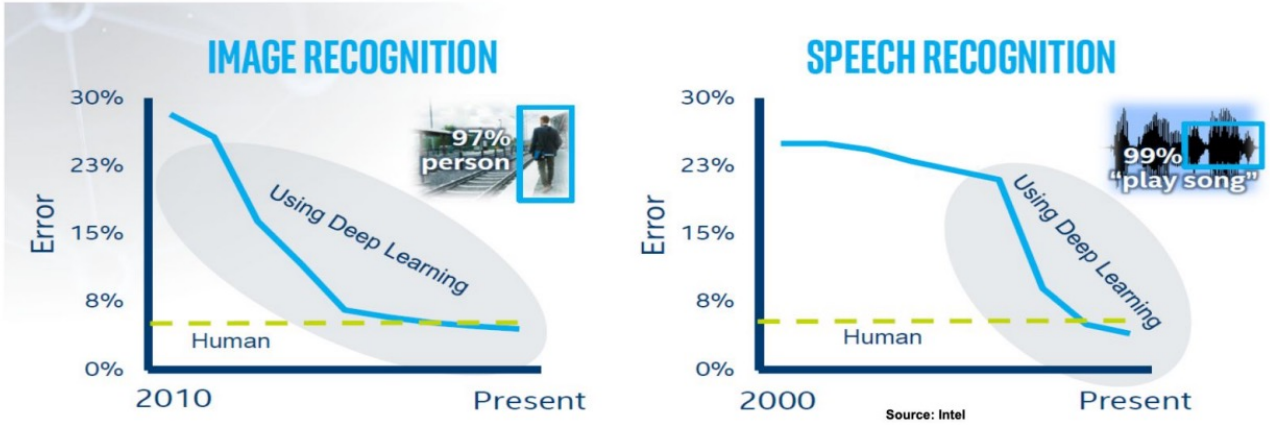


Graph Neural Networks

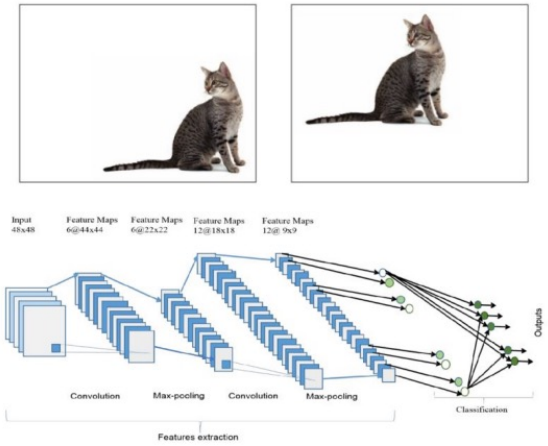


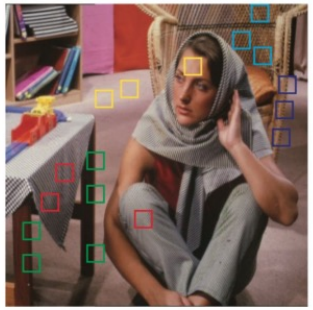
Convolutional Neural Networks (CNNs)

- CNNs' BIG impact!



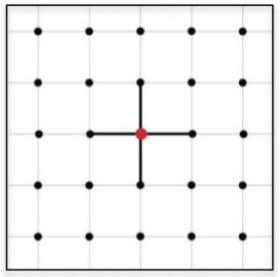
- Key properties of CNNs:
  - Translation Invariance
  - Localized filters in space
  - Multiple Layers



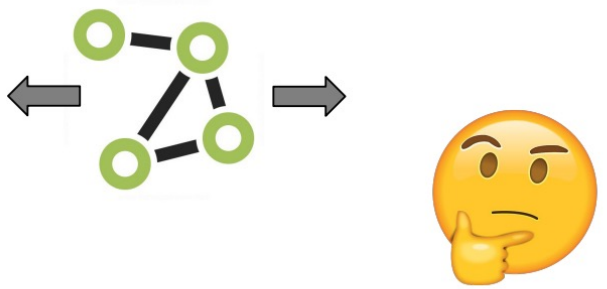


How CNNs for Graphs?

- Translation



$$f(x+a, y+b)$$

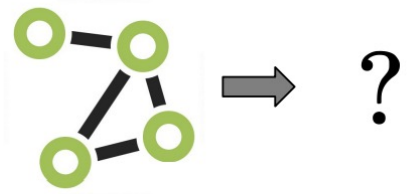


- Downsampling (Pooling)

2	1	3	1
1	0	1	**4**
0	6	**9**	5
7	1	4	1

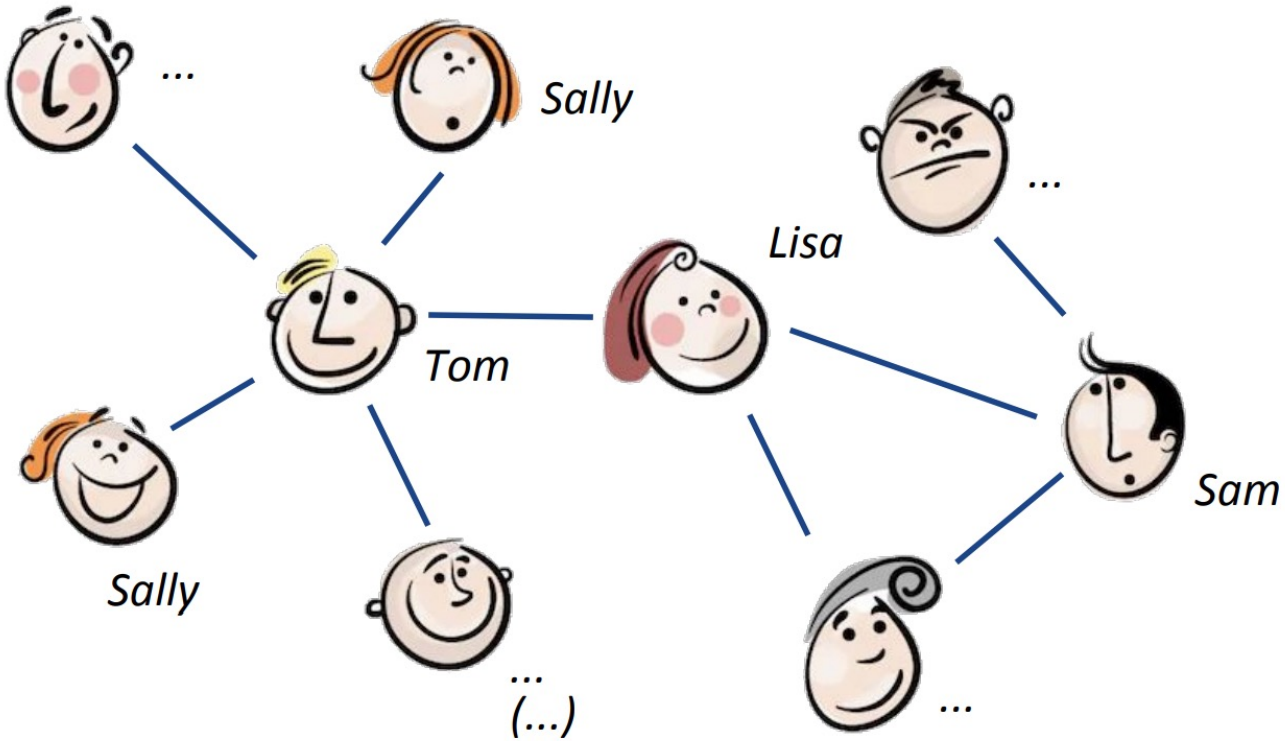
Max Pool

2	4
7	9



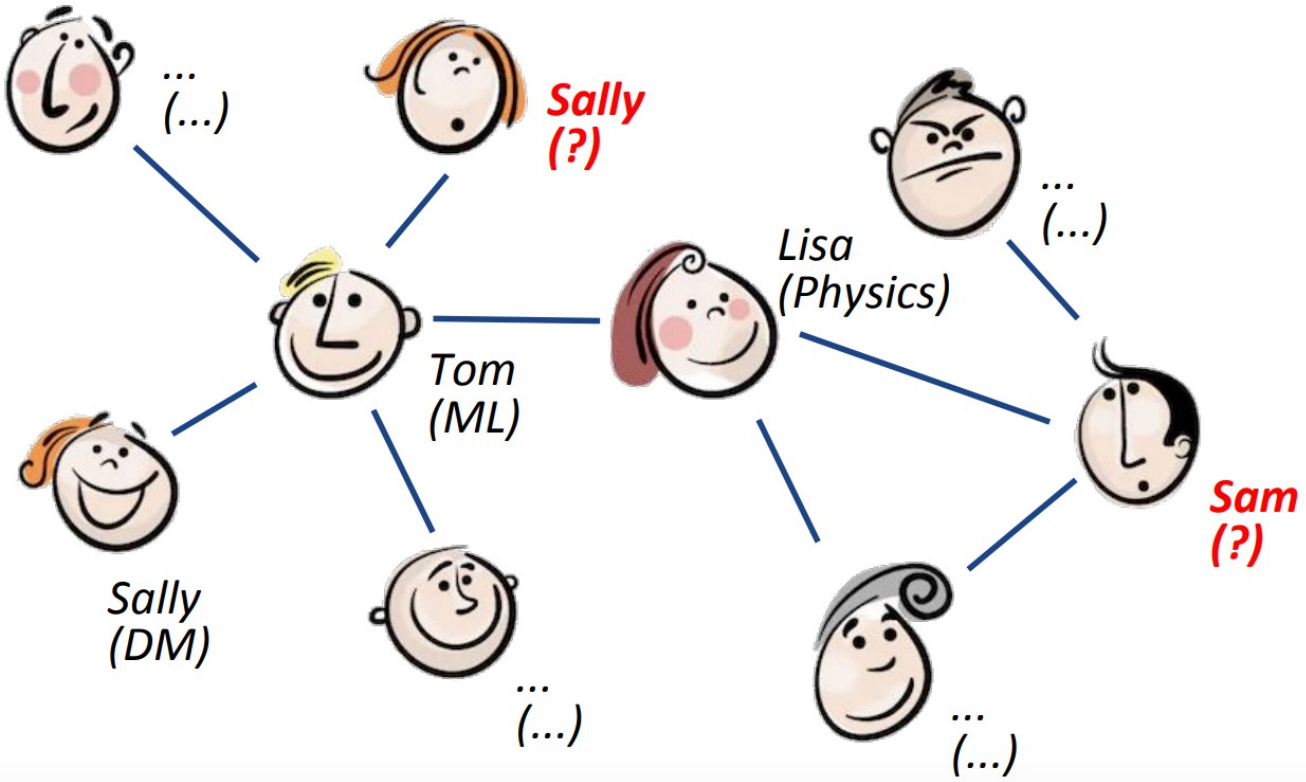
Motivating Example

- **Co-authorship Network**
  - **Nodes**: Authors, **Edges**: Co-authorship



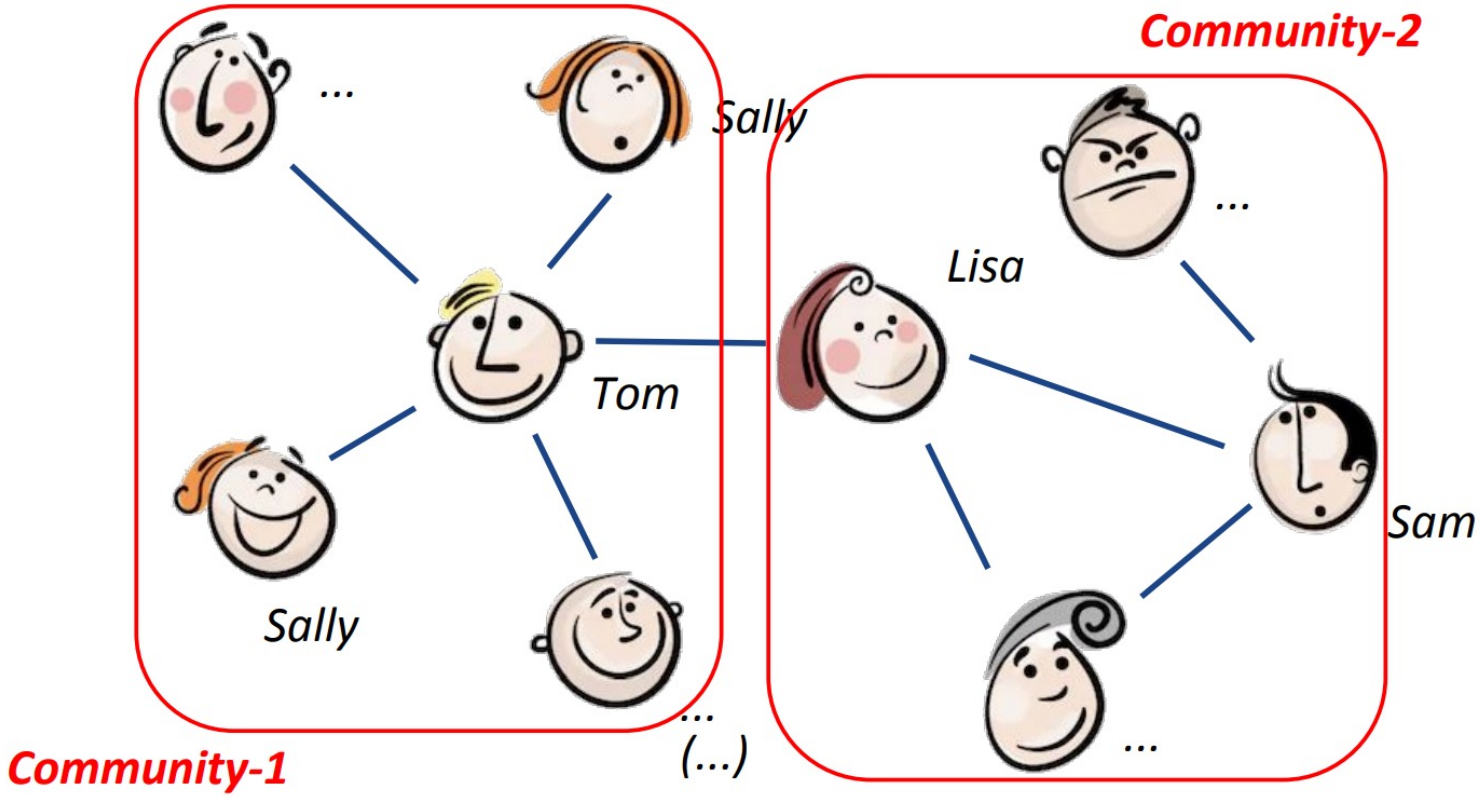
Motivating Example

- **Node Classification:** (Semi-supervised Learning)
  - Predict research area of unlabeled authors



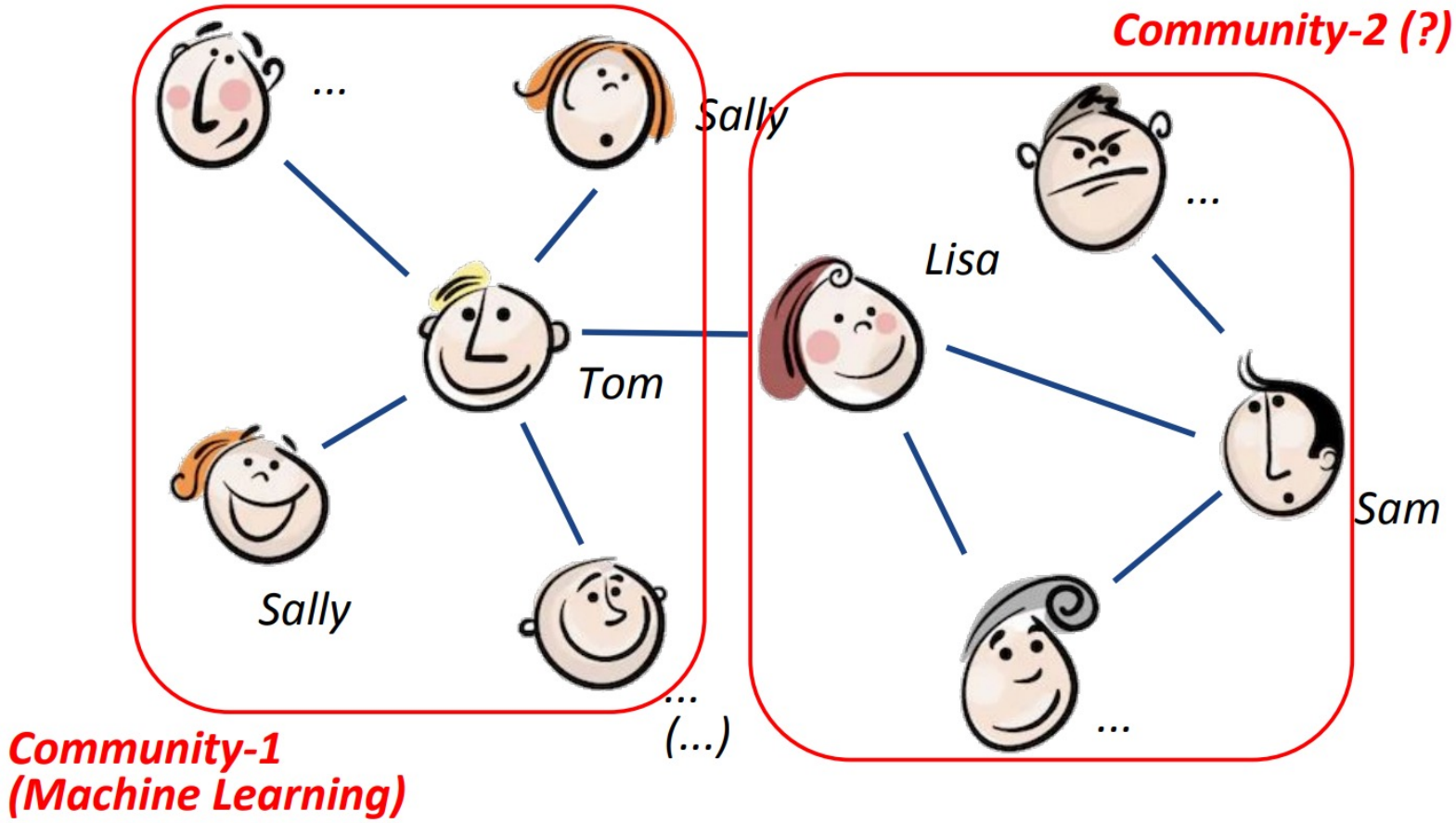
Motivating Example

- **Identify Communities:** (Unsupervised)
  - Grouping authors with similar research interests



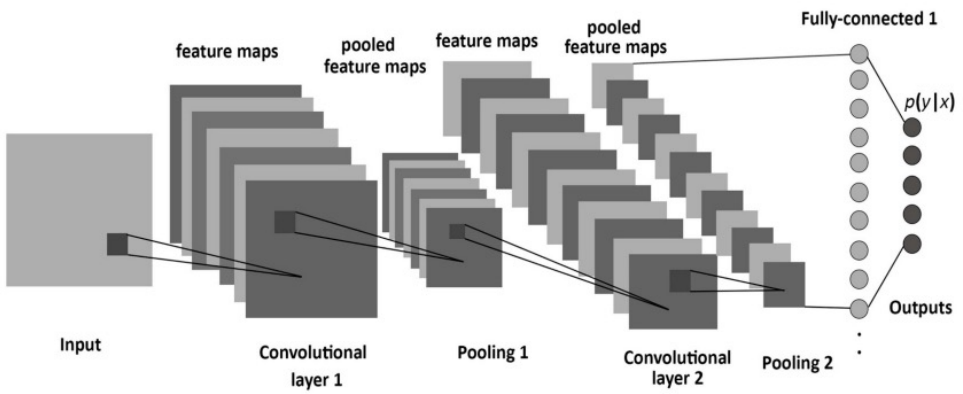
Motivating Example

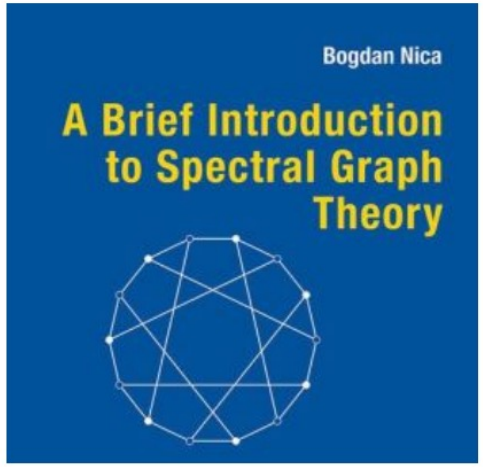
- **Graph Classification:** (Supervised)
  - Identifying class of each community.



Graph Convolutional Networks (GCN)

- GCN formulation





$$h_v = f\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} W x_u + b\right), \quad \forall v \in \mathcal{V}.$$

Graph Convolutional Networks (GCN)

GNN formulation by [Kipf et al., ICLR 2016]

$$h_v = f\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} W x_u + b\right), \quad \forall v \in \mathcal{V}.$$

Filter matrix (Model-parameter)

Initial Node Features

Nodes == Words → Word2vec Embeddings
Nodes == Authors → 0/1 value indicating frequently used keywords
No features → One-hot vector (length = #Nodes)

Graph Convolutional Networks (GCN)

GNN formulation by [Kipf et al., ICLR 2016]

$$h_v = f\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} W x_u + b\right), \quad \forall v \in \mathcal{V}.$$

Normalization

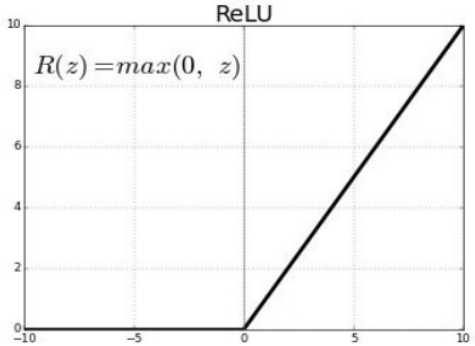
Neighborhood Aggregation

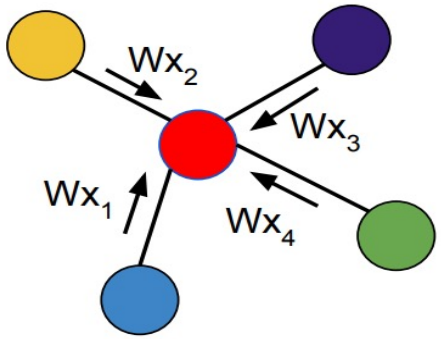
Bias (Model-parameter)

Filter matrix (Model-parameter)

Initial Node Features

Non-Linearity





Graph Convolutional Networks (GCN)

GNN formulation by [Kipf et al., ICLR 2016]

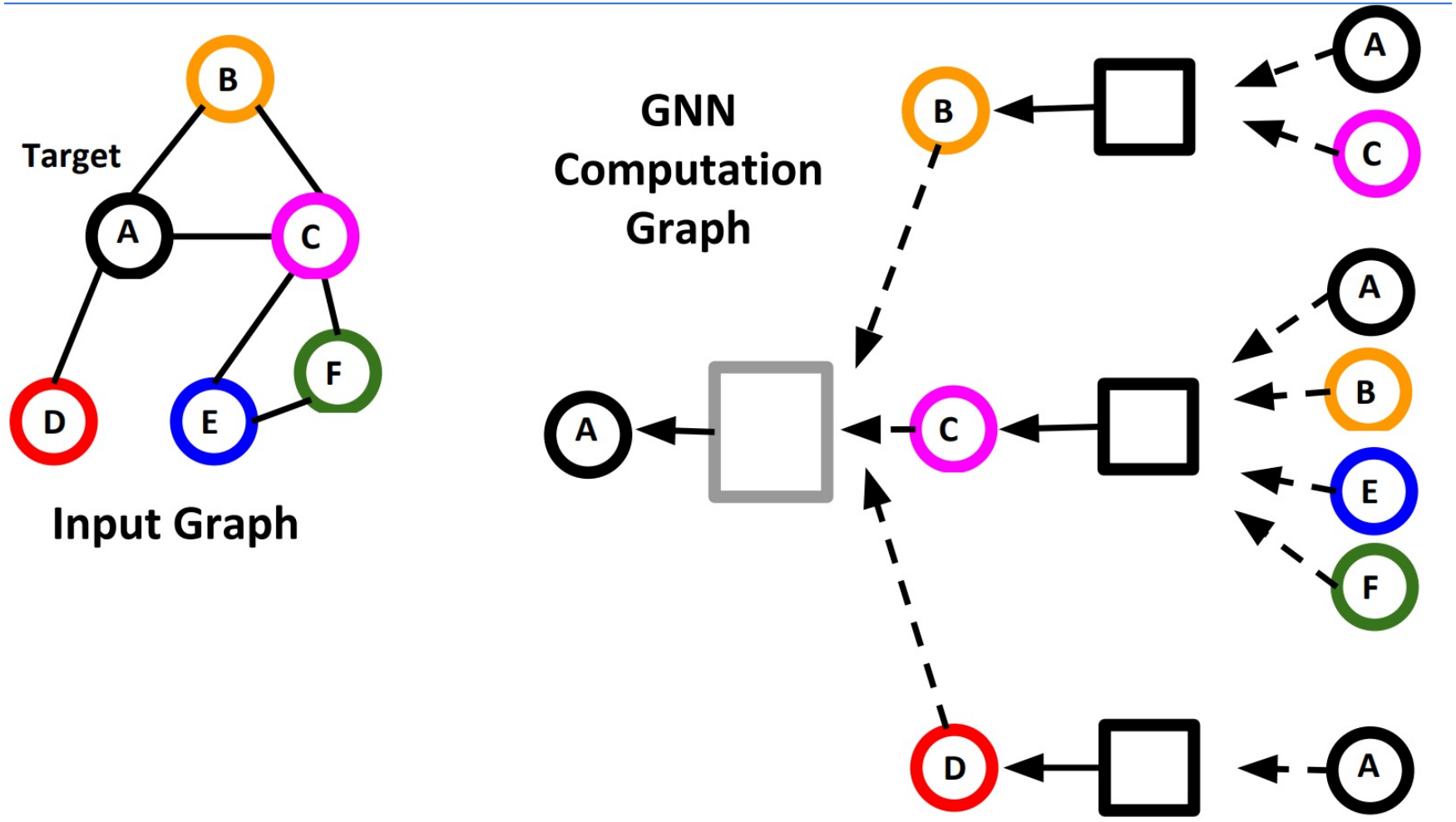
$$h_v = f\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} W x_u + b\right), \quad \forall v \in \mathcal{V}.$$

The above formulation is restricted to capturing just 1-hop of nodes

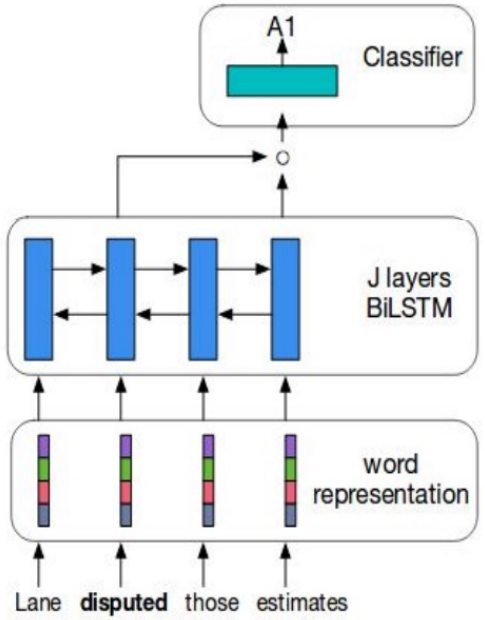
Stacking K-GNN Layers for capturing K-hop nbd

$$h_v^{k+1} = f\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} W^k h_u^k + b^k\right), \quad \forall v \in \mathcal{V}.$$

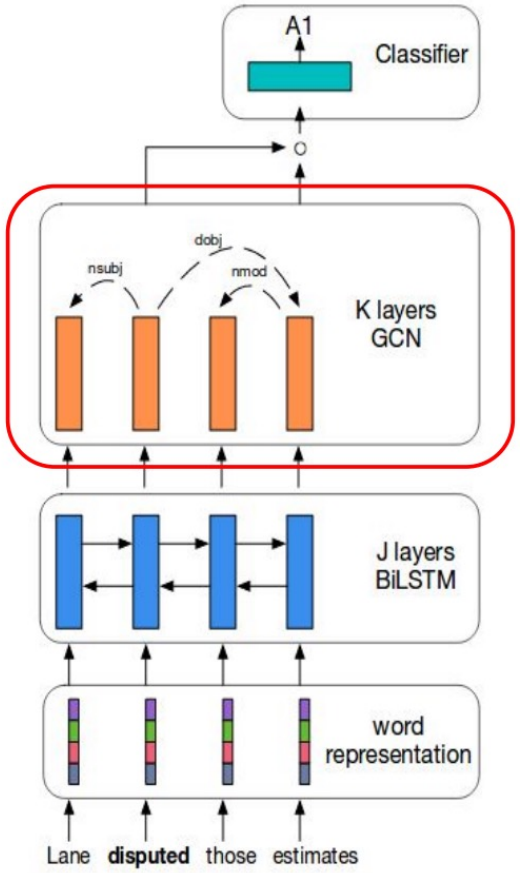
Graph Convolutional Networks (GCN)



Example: GNNs for Semantic Role Labeling



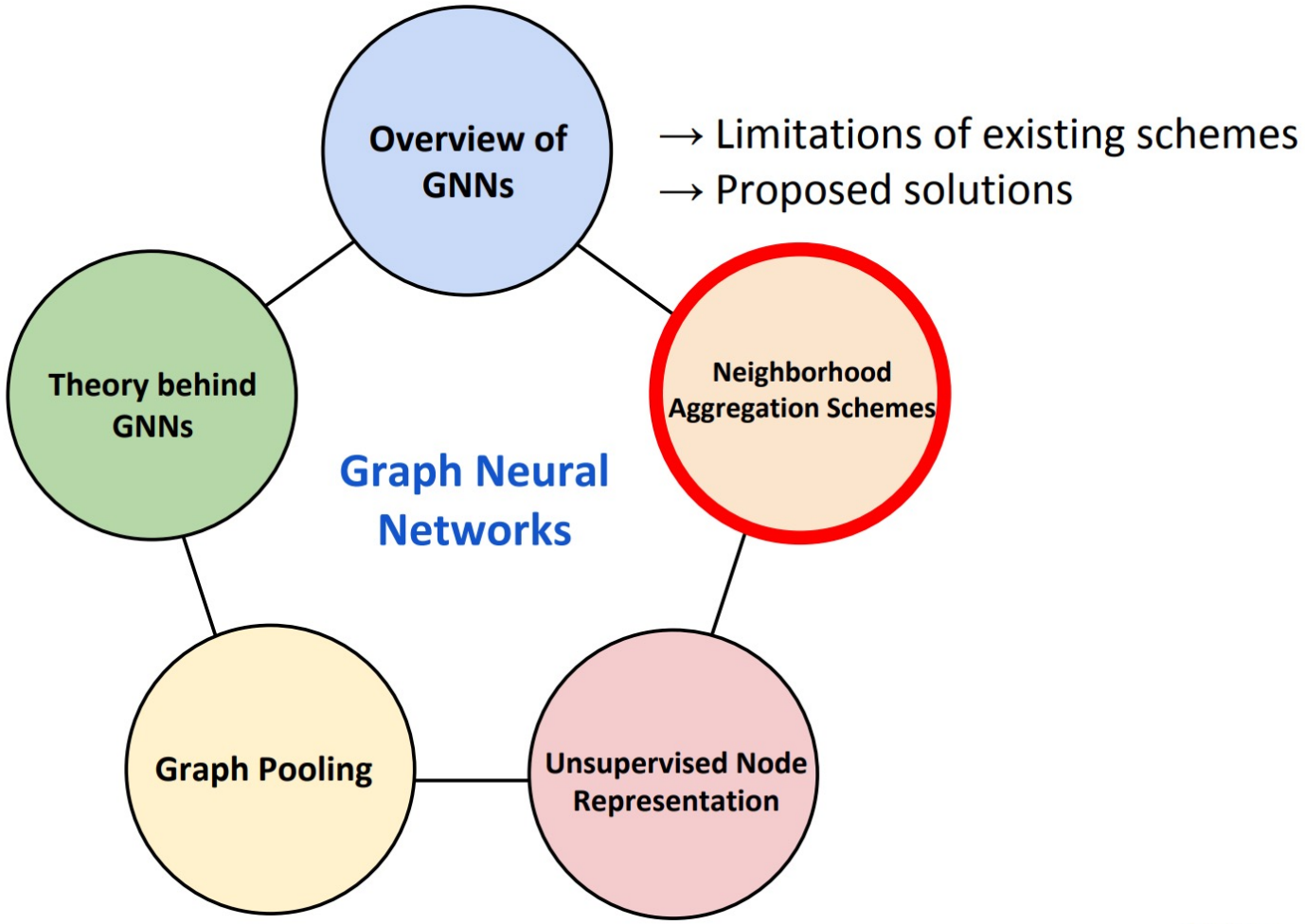
Standard Deep Learning Architecture for NLP problems (above is for Semantic-Role Labeling (SRL))



Model with GCN as part of the network

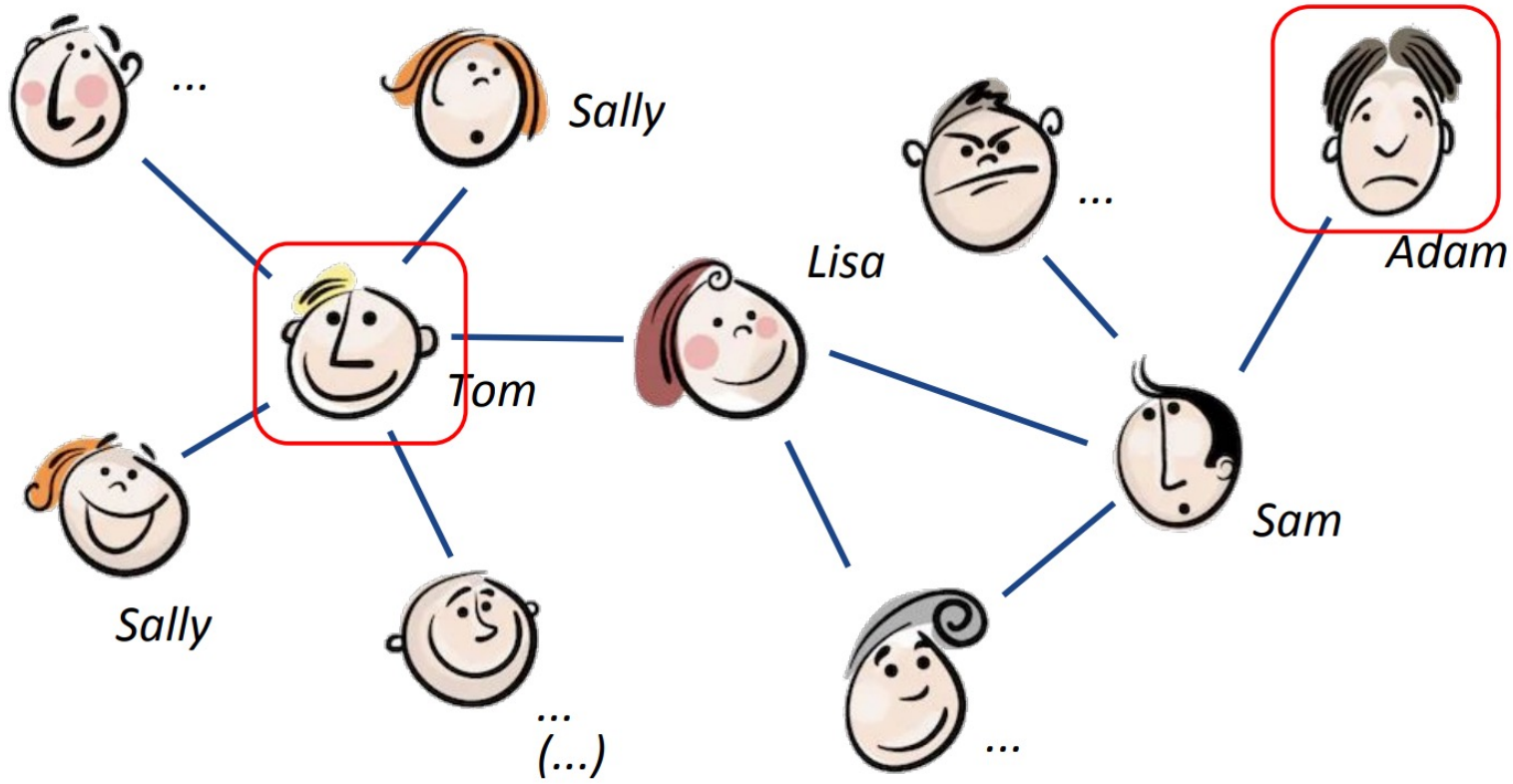
GCN weights are trained based on the final objective

Graph Neural Networks



Motivating Example: Co-authorship Network

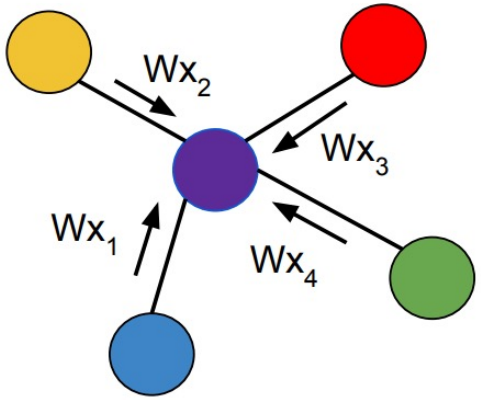
- Handling heterogeneous graphs



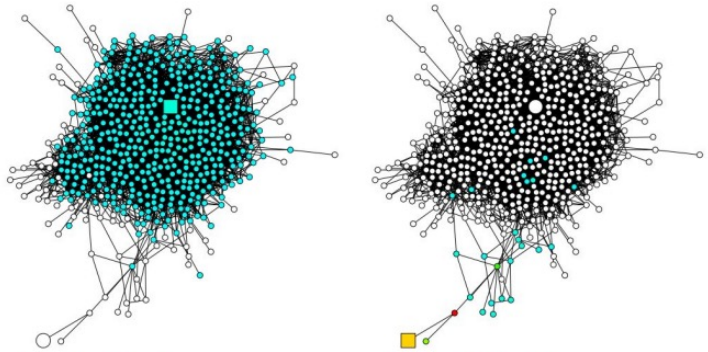
Neighborhood Aggregations in GCNs

Standard GCN neighborhood aggregation

$$h_v = f\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} W x_u + b\right), \quad \forall v \in \mathcal{V}.$$



No restriction on influence neighborhood



Hub Node Leaf Node

Methods:

- Graph Attention Networks (GAT)
- Confidence-based GCN (ConfGCN)

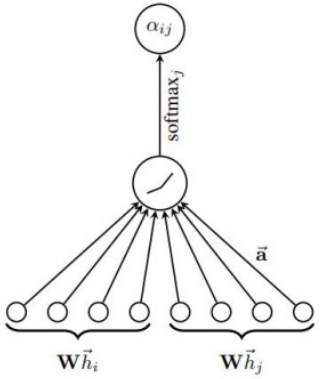
Graph Attention Networks [Veličković et al., ICLR'18]

Uses self-attention for GCNs

Input features: $\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F,$

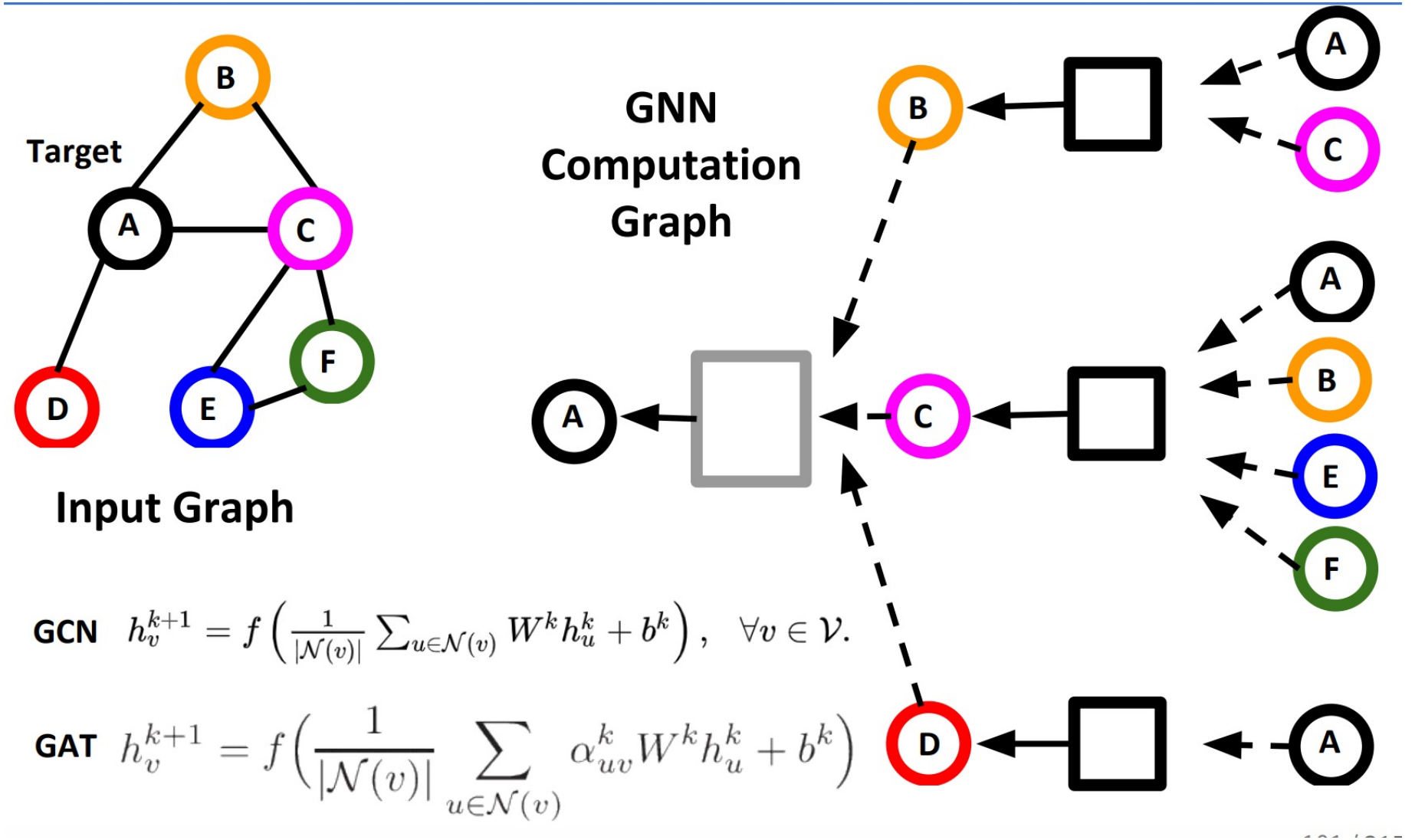
Importance of v_j to v_i: $e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j) \quad \alpha_{ij} = \mathrm{softmax}_j(e_{ij})$

$$\alpha_{ij} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$



Method	Cora	Citeseer	Pubmed
GCN	81.5%	70.3%	**79.0%**
GAT	**83.0** ± 0.7%	**72.5** ± 0.7%	**79.0** ± 0.3%

GCN vs GAT

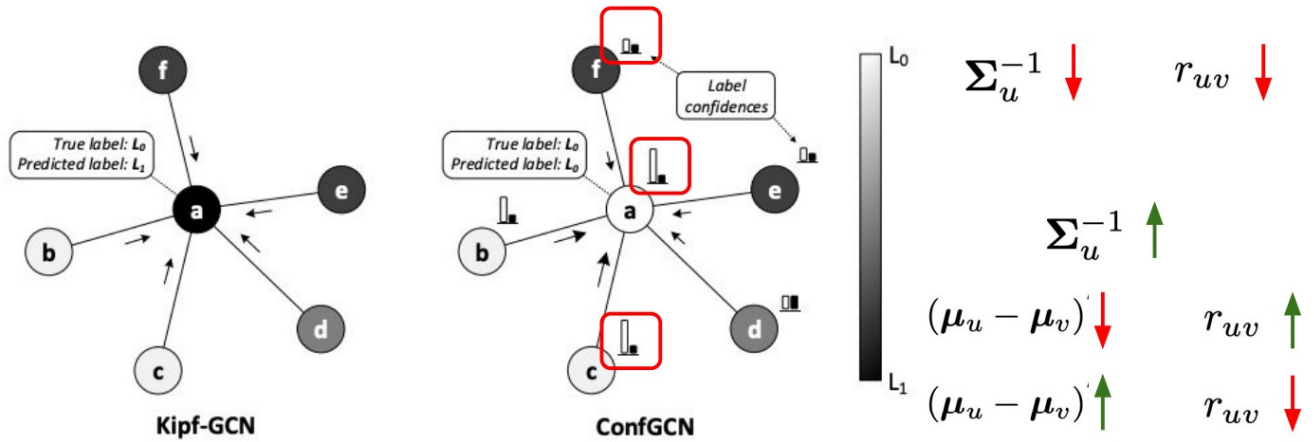


GCN $h_v^{k+1} = f\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} W^k h_u^k + b^k\right), \quad \forall v \in \mathcal{V}.$

GAT $h_v^{k+1} = f\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} \alpha_{uv}^k W^k h_u^k + b^k\right)$

Confidence-based GCN [Vashishth et al., AISTATS'19]

- Comparison with standard GCN model

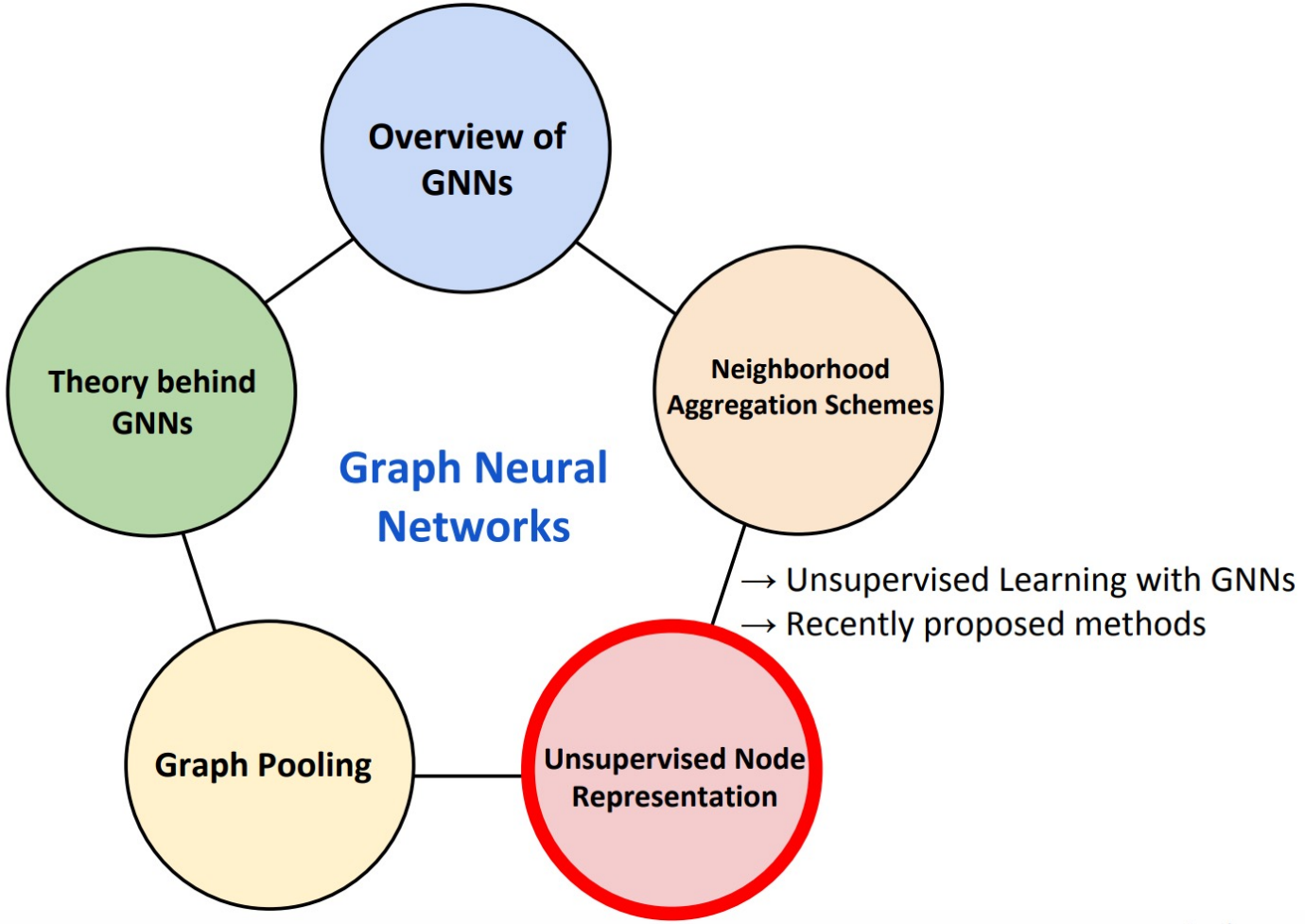


- Importance for a node is calculated as:

$$r_{uv} = \frac{1}{d_M(u,v)}. \quad d_M(u,v) = (\boldsymbol{\mu}_u - \boldsymbol{\mu}_v)^T(\boldsymbol{\Sigma}_u^{-1} + \boldsymbol{\Sigma}_v^{-1})(\boldsymbol{\mu}_u - \boldsymbol{\mu}_v).$$

  - μ_u, μ_v are label distribution and Σ_u, Σ_v denote co-variance matrices.

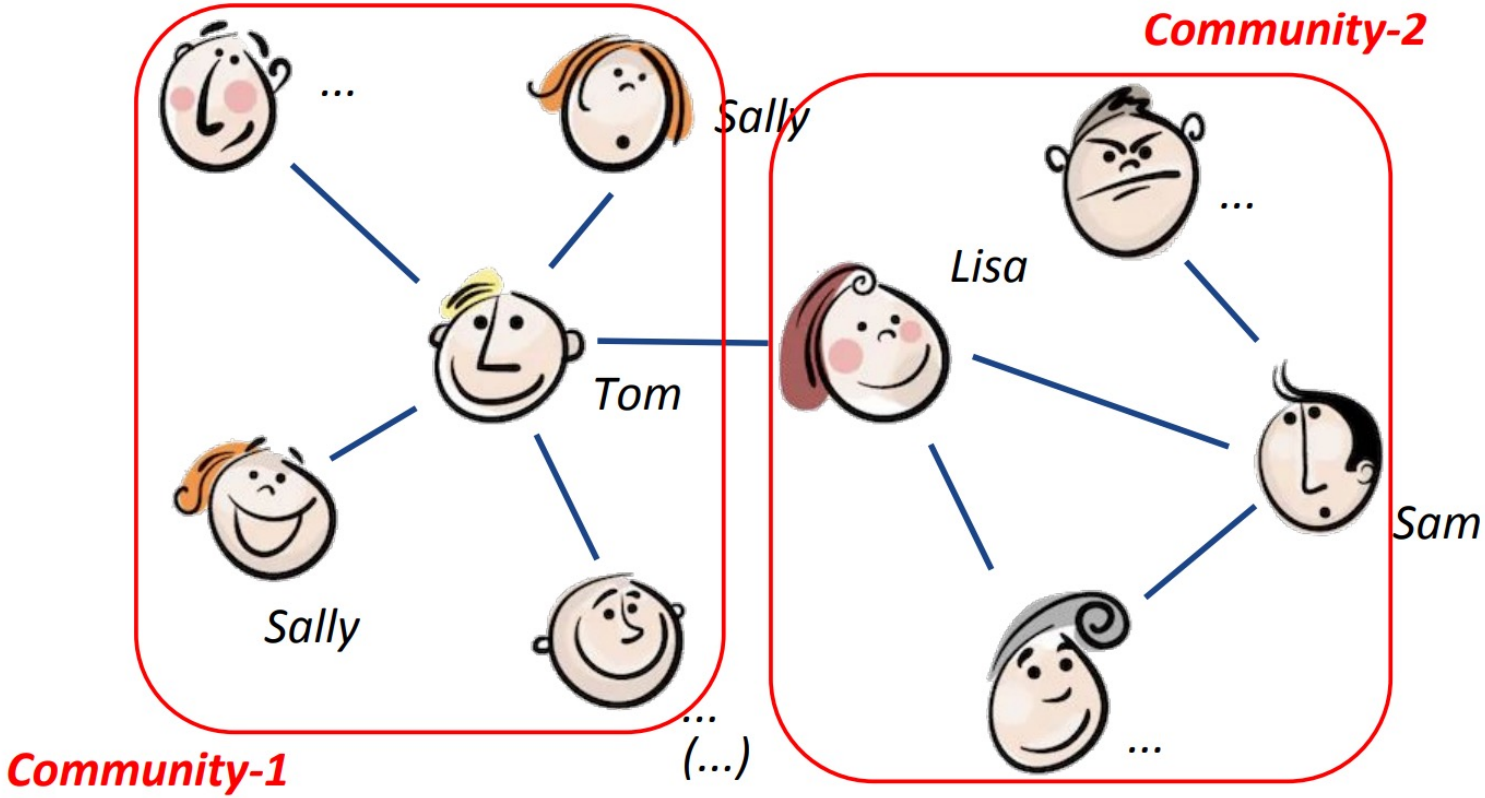
Graph Neural Networks



Motivating Example

Identify Communities: (Unsupervised)

- Grouping authors with similar research interests



Unsupervised Representation Learning

Labeled data is expensive

Allows to discover interesting structure from large-scale graphs

Methods

- GraphSAGE
- Graph Auto-Encoder (GAE)
- Deep Graph Infomax (DGI)

GraphSAGE [Hamilton et al., NeurIPS'17]

Propose three neighborhood aggregators

Mean Aggregator:

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}).$$

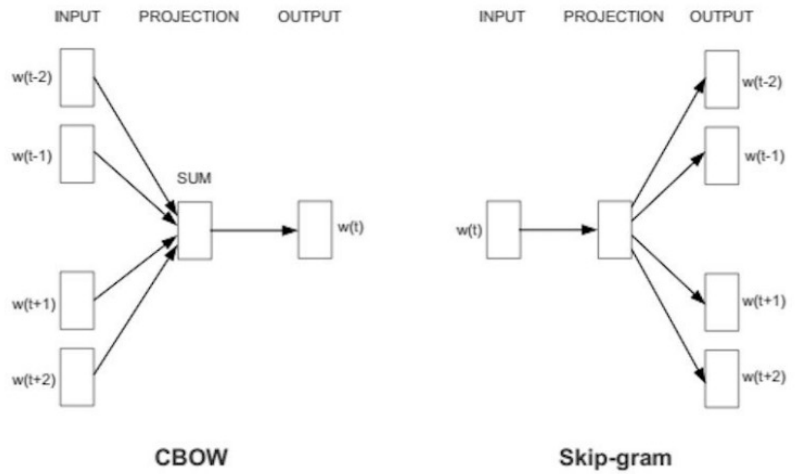
LSTM Aggregator:

Applies LSTM to a random permutation of neighbors.

Pooling Aggregator:

$$\text{AGGREGATE}_k^{\text{pool}} = \max(\{\sigma\left(\mathbf{W}_{\text{pool}}\mathbf{h}_{u_i}^k + \mathbf{b}\right), \forall u_i \in \mathcal{N}(v)\}),$$

GraphSAGE



$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\right),$$

Nearby nodes →

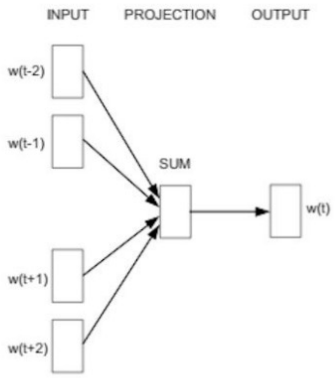
Similar representation

v: *is a node that co-occurs near* ***u***

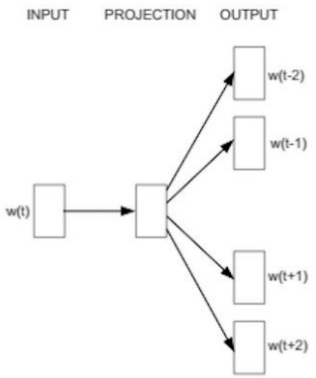
P_n: *Negative sampling distribution (from word2vec)*

z_u: *Embedding from GraphSAGE*

GraphSAGE



CBOW



Skip-gram

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\right),$$

Nearby nodes → Similar representation

Disparate nodes → Highly distinct representation

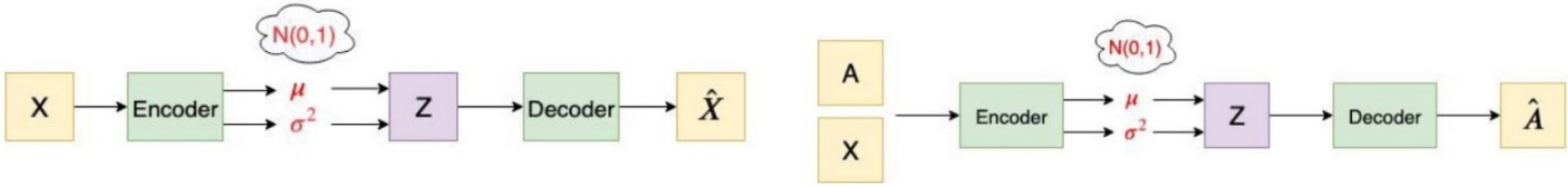
v: *is a node that co-occurs near* ***u***

P_n: *Negative sampling distribution (from word2vec)*

z_u: *Embedding from GraphSAGE*

Graph Auto-Encoder (GAE) [Kipf et al., BDL-NeurIPS '16]

- Variational autoencoder (VAE) based model



Encoder:

$$\mu = GCN_{\mu}(X, A)$$
$$\sigma = GCN_{\sigma}(X, A),$$
$$GCN(X, A) = \hat{A}ReLU(\hat{A}XW_0)W_1$$

Decoder:

$$p(\mathbf{A} \mid \mathbf{Z}) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(A_{ij} \mid \mathbf{z}_i, \mathbf{z}_j) \ ,$$
$$\text{with} \quad p(A_{ij} = 1 \mid \mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^{\top} \mathbf{z}_j) \ ,$$

Loss:

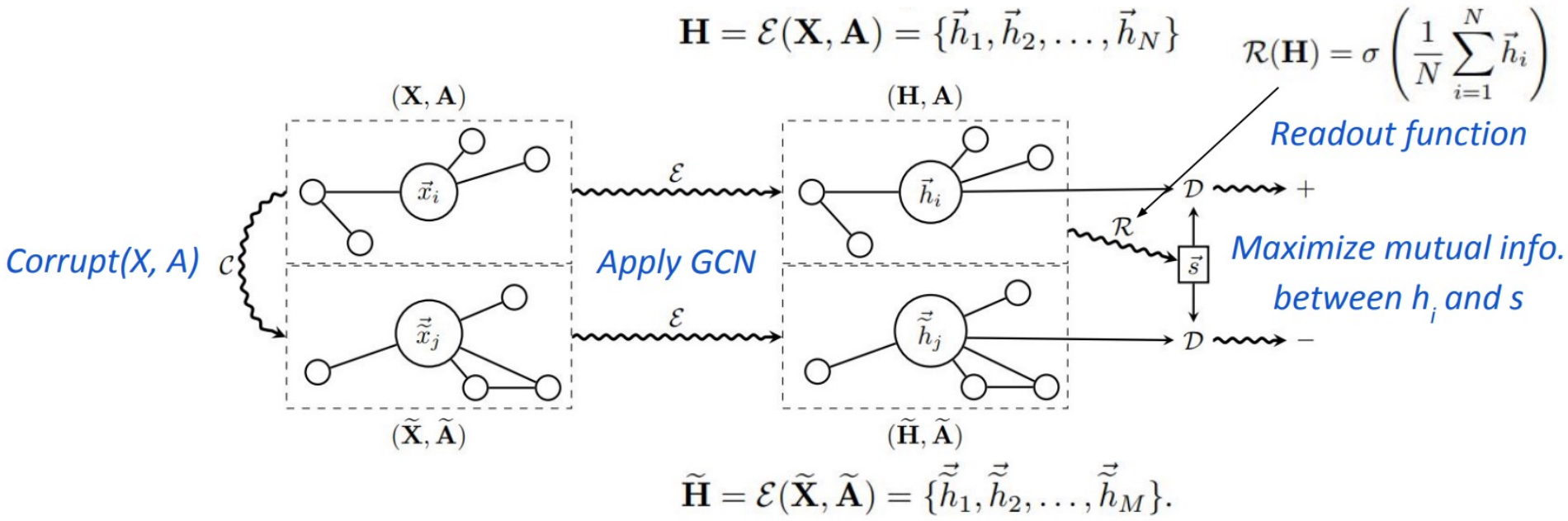
$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}\left[\log p(\mathbf{A} \mid \mathbf{Z})\right] - \mathrm{KL}\left[q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) \,||\, p(\mathbf{Z})\right] \ ,$$

Reconstruction Loss

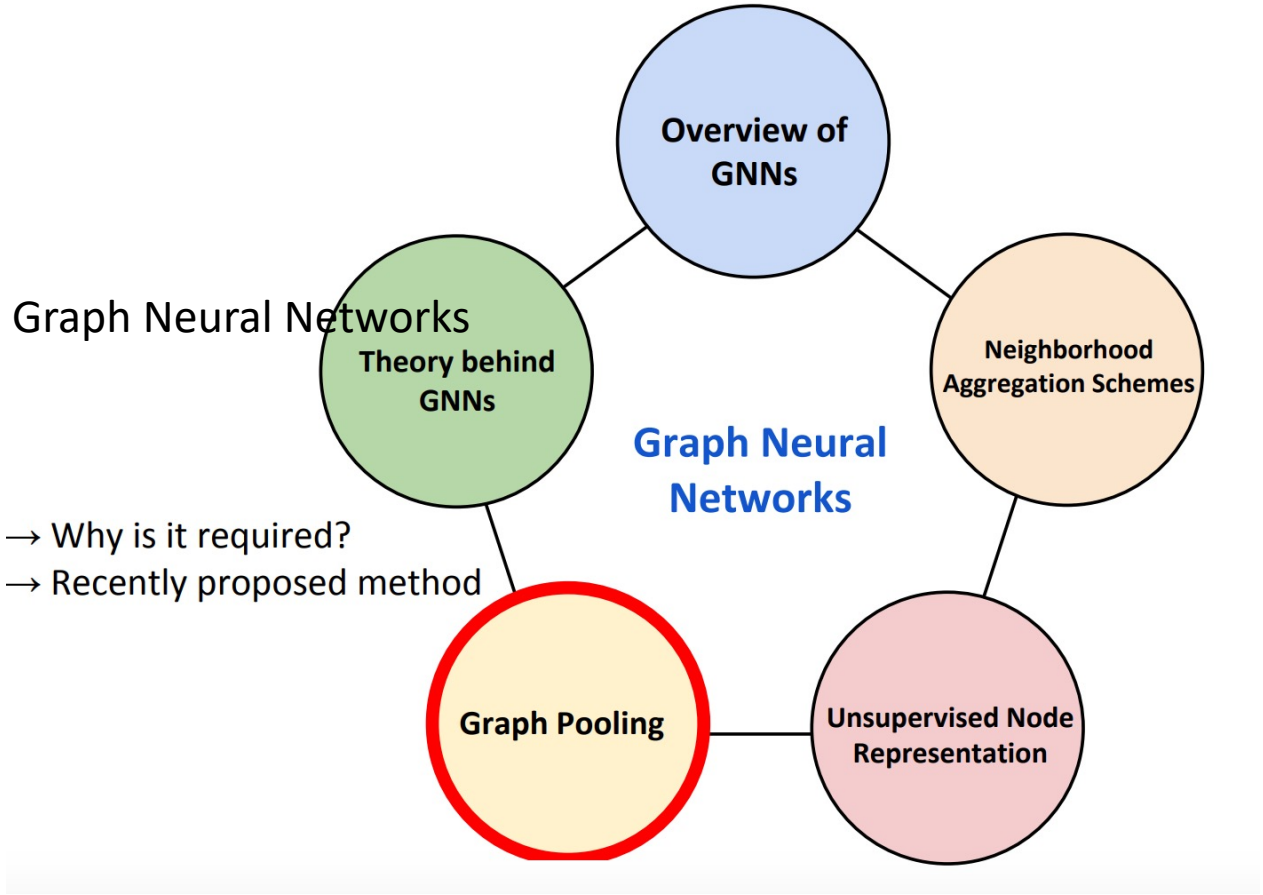
KL-Divergence

Deep Graph Infomax [Velickovic´ et al. ICLR ’19]

- Maximizes mutual information across graph’s patch representations.

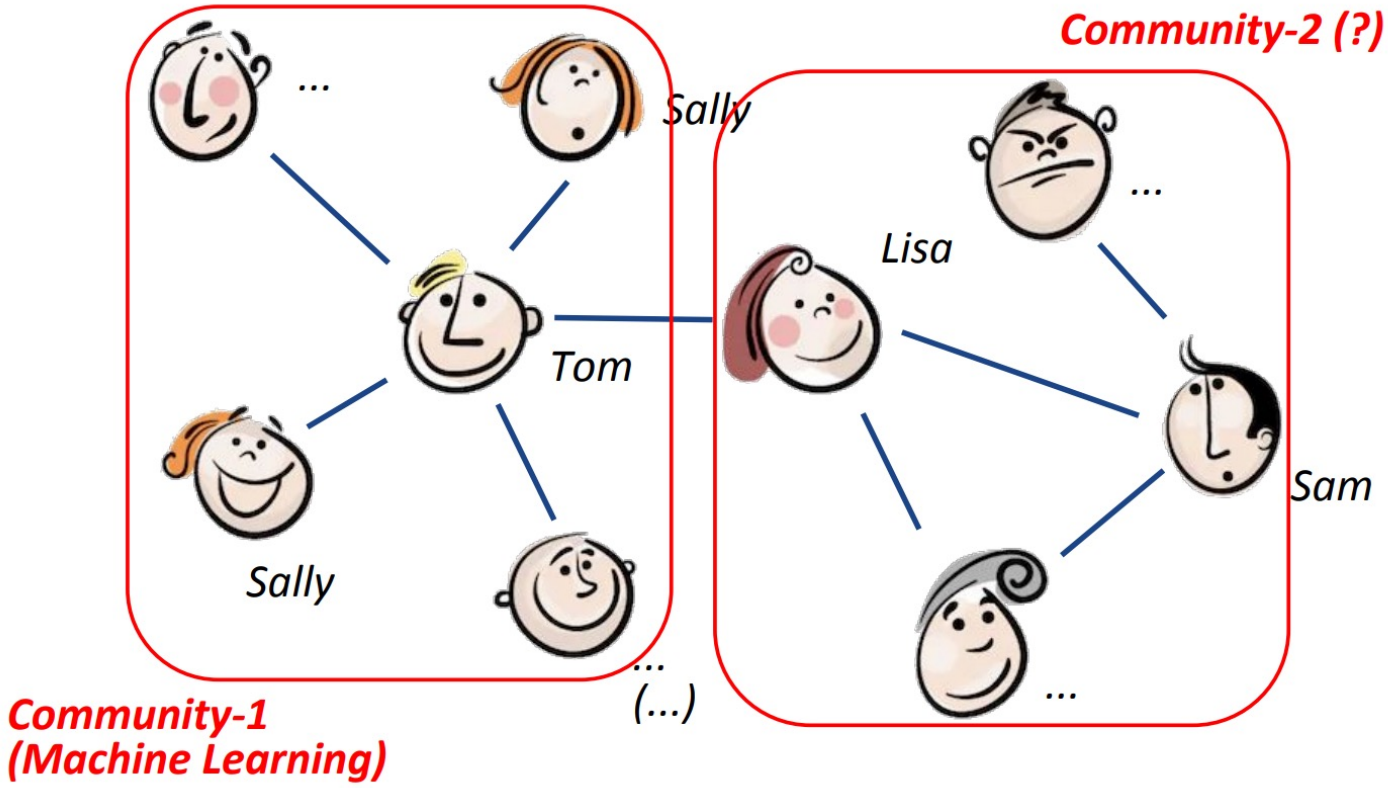


Graph Neural Networks



Motivating Example

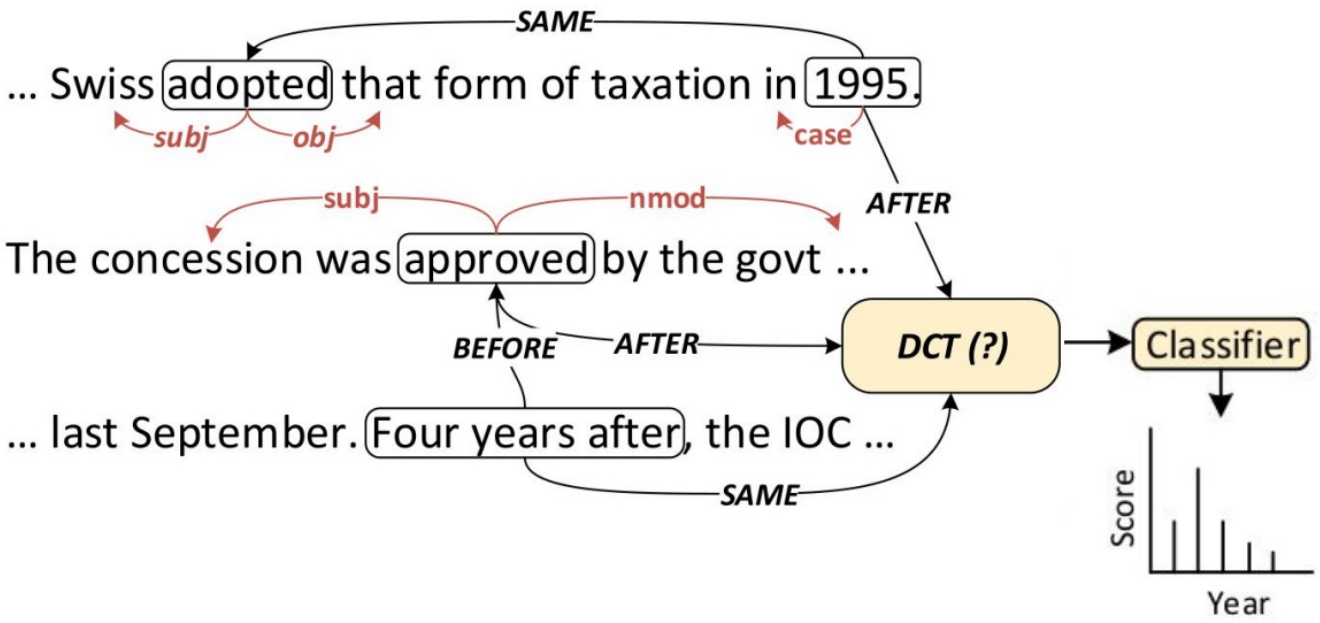
- **Graph Classification:** (Supervised)
  - Identifying class of each community.



Motivating Example from NLP

Document Classification

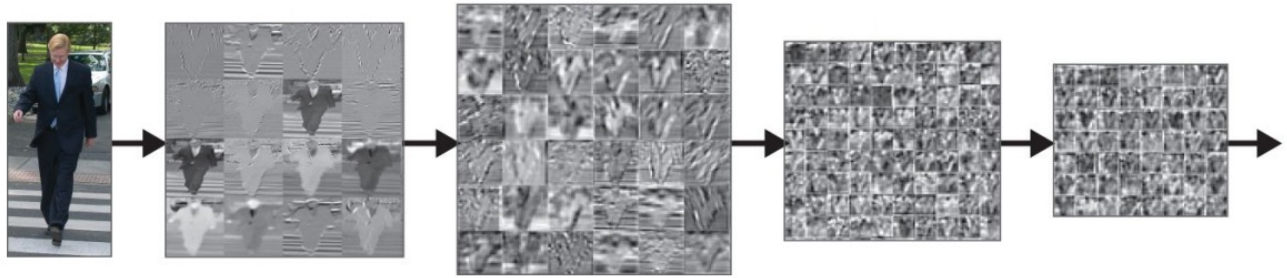
- Getting a representation for the entire document



Graph Pooling

Essential for graph-level tasks (Graph Classification)

Goal: Get an embedding for the entire graph



Graph clustering is NP-Hard.

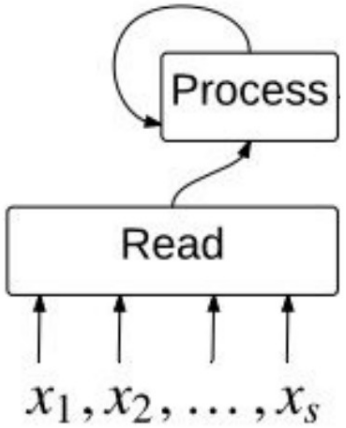
Graph Pooling

Methods

- Simple max/mean pooling
  - Inefficient and overlooks node ordering
- Graclus clustering algorithm
- Set2Set method
- Differentiable Pooling (DiffPool)

Graph Pooling: Set2Set [Vinyals et al., ICLR'15]

- Set2Set has three components:
  - Reading block: Embeds each element onto a memory unit.
  - Process block: Performs T steps of LSTM w/o input/output



Assumes an ordering on all the vertices

Takes random initial state q_0

$$q_t = LSTM(q^*_{t-1}) \quad (3)$$

$$e_{i,t} = f(m_i, q_t) \quad (4)$$

$$a_{i,t} = \frac{\exp(e_{i,t})}{\sum_j \exp(e_{j,t})} \quad (5)$$

Computes attention scores over all nodes

$$r_t = \sum_i a_{i,t} m_i \quad (6)$$

$$q^*_t = [q_t \; r_t] \quad (7)$$

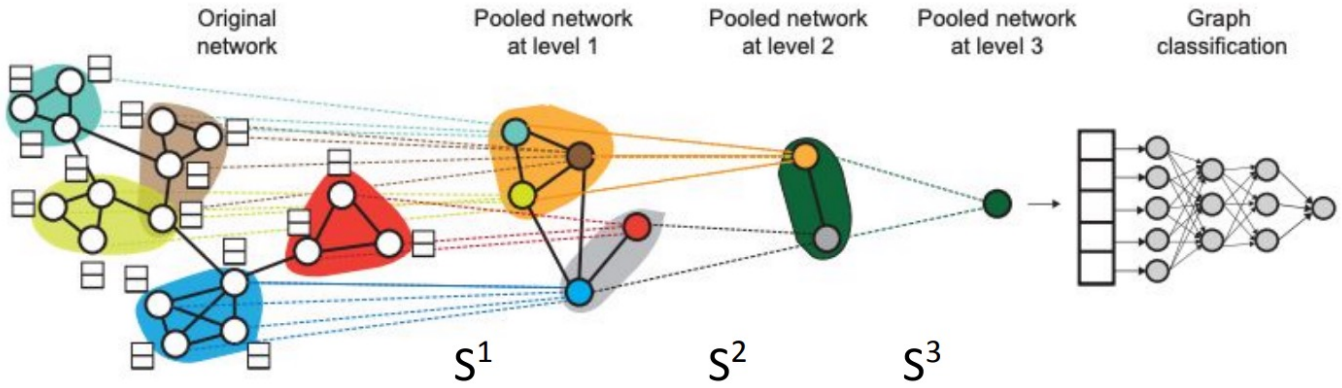
Computes weighted sum and pass it back to LSTM

q_T is the graph representation which is invariant of initial vertex ordering

Graph Pooling: DiffPool [Ying et al. NeurIPS'18]

- Learns differentiable soft cluster assignment (S^l) from nodes in layer l to nodes in layer (l+1)

$$S^l \in \mathbb{R}^{n_l \times n_{l+1}}$$



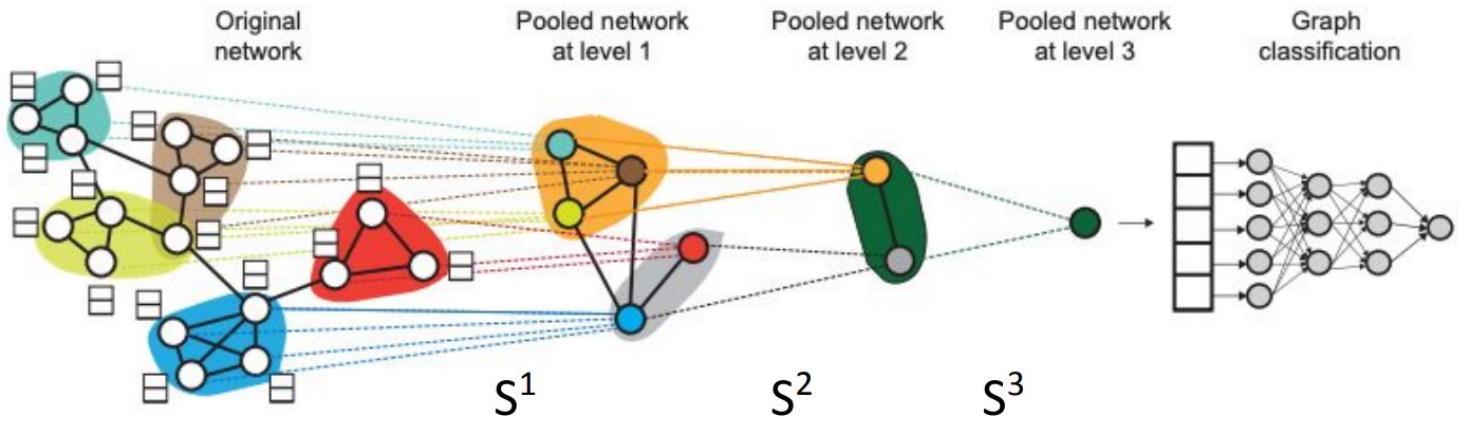
$$S^{(l)} = \text{softmax}\left(\text{GNN}_{l,\text{pool}}(A^{(l)}, X^{(l)})\right) \qquad X^{(l+1)} = S^{(l)^T} Z^{(l)} \in \mathbb{R}^{n_{l+1} \times d},$$

$$Z^{(l)} = \text{GNN}_{l,\text{embed}}(A^{(l)}, X^{(l)}), \qquad A^{(l+1)} = S^{(l)^T} A^{(l)} S^{(l)} \in \mathbb{R}^{n_{l+1} \times n_{l+1}}.$$

Graph Pooling: DiffPool

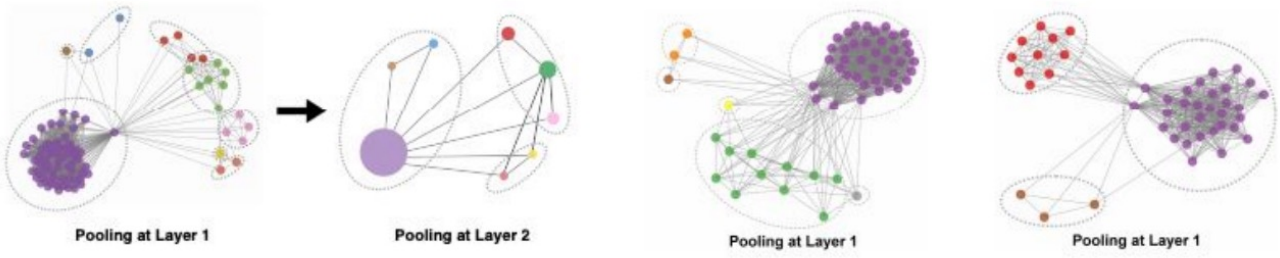
- Learns differentiable soft cluster assignment (S^l) from nodes in layer l to nodes in layer (l+1)

$$S^l \in \mathbb{R}^{n_l \times n_{l+1}}$$



$$(A^{(l+1)}, X^{(l+1)}) = \text{DIFFPOOL}(A^{(l)}, Z^{(l)})$$

Graph Pooling: DiffPool



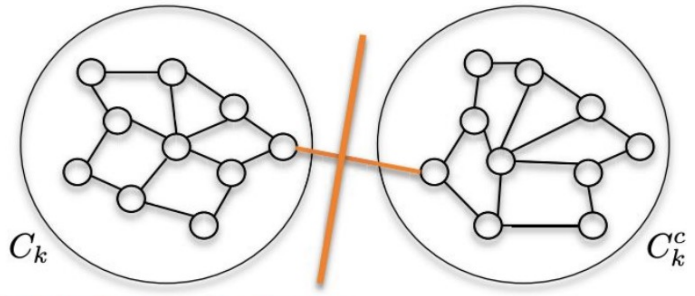
Learned clusters are interpretable

	Method	ENZYMES	D&D
GNN	GRAPHSAGE	54.25	75.42
	ECC	53.50	74.10
	SET2SET	60.15	78.12
	SORTPOOL	57.12	79.37
	DIFFPOOL-DET	58.33	75.47
	DIFFPOOL-NOLP	61.95	79.98
	DIFFPOOL	**62.53**	**80.64**

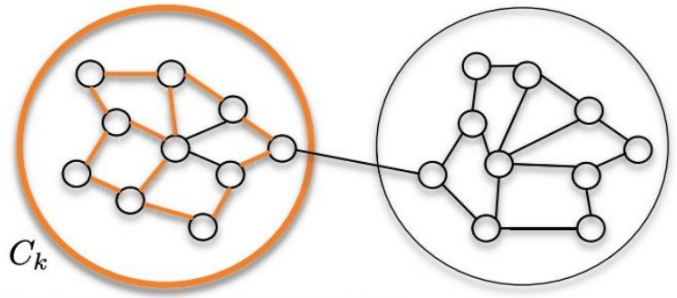
Outperforms existing methods

Appendix

Graph Clustering Measures



Partitioning by min edge cuts.



Partitioning by max vertex matching.

Normalized Cut:

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \frac{\mathrm{Cut}(C_k, C_k^c)}{\mathrm{Vol}(C_k)}$$

⇕ Equivalence by complementarity

Normalized Association:

$$\max_{C_1,\ldots,C_K} \sum_{k=1}^{K} \frac{\mathrm{Assoc}(C_k)}{\mathrm{Vol}(C_k)}$$

where

$$Cut(A,B) = \sum_{i\in A, j\in B} W_{ij}$$

$$Vol(A) = \sum_{i\in A} d_i$$

$$Assoc(A) = \sum_{i\in A, j\in A} W_{ij}$$

Graph Pooling: Graclus Clustering

- Greedy algorithm, minimizes Normalized Association.
  - Vertex Matching
  - Graph Coarsening