# Graph Edit Distance Computation via Graph Neural Networks

Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, Wei Wang

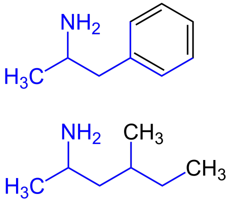
Presenter: Jack Lanchantin
`https://qdata.github.io/deep2Read`

## Outline

## Outline

## Graph Similarity Matching

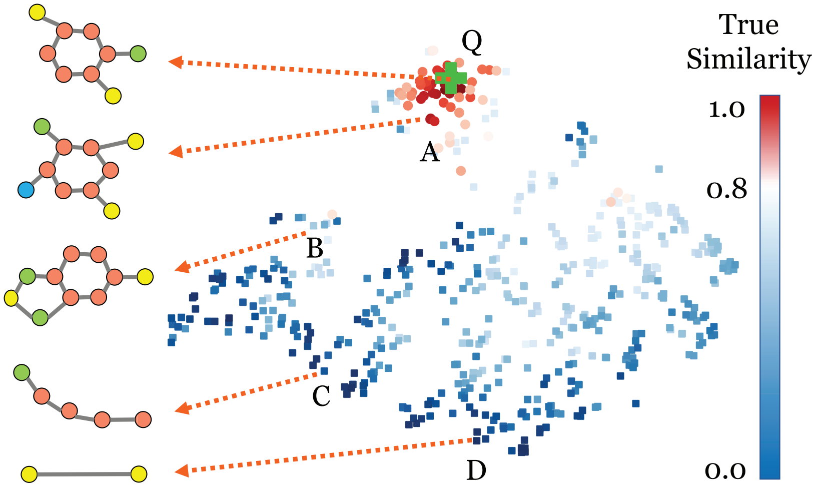
Graph similarity: finding a graph most similar to query graph (e.g. finding the chemical compounds similar to query compound).



## SimGNN

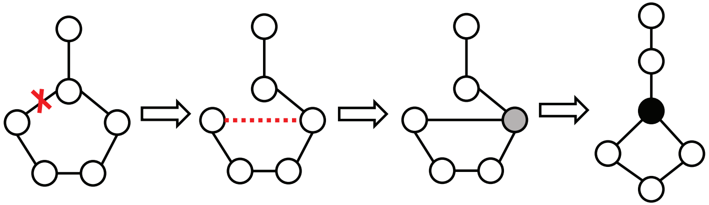
- Graph similarity/distance computation, such as Graph Edit Distance and Maximum Common Subgraph, are the core operation of graph similarity search, but very costly to compute
- The proposed approach, called SimGNN, uses two steps
  - First, a learnable embedding function is used to map every graph into an embedding vector for each node (or a global embedding vector)
  - Second, a graph similarity function is used to compare the two graphs

## Task



## Outline

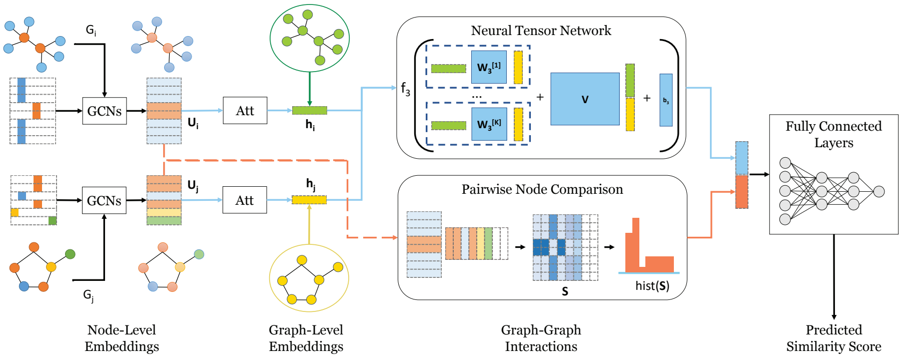
## Graph Edit Distance



The GED between the graph to the left and the graph to the right is 3, as the transformation needs 3 edit operations: (1) an edge deletion, (2) an edge insertion, and (3) a node relabeling.

# Outline

# SimGNN



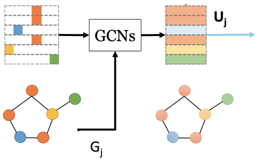
$G_i$
GCNs
$U_i$
Att
$h_i$
$U_j$
$h_j$
$G_j$
Neural Tensor Network
$f_3$
$W_3^{[1]}$
...
$W_3^{[K]}$
+
V
$b_3$
Pairwise Node Comparison
S
hist(S)
Fully Connected Layers
Node-Level Embeddings
Graph-Level Embeddings
Graph-Graph Interactions
Predicted Similarity Score

## Stage I: Node Embedding

- 3 Layer GCN to produce node embeddings
- Nodes, denoted by $\boldsymbol{u_n} \in \mathbb{R}^D$, are updated as follows:

$$\operatorname{conv}(\boldsymbol{u_n}) = \sigma\left( \sum_{m \in \mathcal{N}(n)} \frac{1}{\sqrt{d_n d_m}} \boldsymbol{u_m} \boldsymbol{W_1}^{(l)} \right) \tag{1}$$

$\mathcal{N}(n)$ is the first-order neighbors of node $n$, $d_n$ is the degree of node $n$, $\boldsymbol{W_1}^{(l)} \in \mathbb{R}^{D^l \times D^{l+1}}$ is the weight matrix for the $l$-th GCN layer



## Stage II: Graph Embedding: Global Context-Aware Attention

- Given node embeddings $\boldsymbol{u_n} \in \mathbb{R}^D$, global graph context $\boldsymbol{c} \in \mathbb{R}^D$ is computed, which is an average of node embeddings followed by tanh:

$$\boldsymbol{c} = \tanh\Big(\Big((\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{u_n}\Big)\boldsymbol{W_2}\Big), \tag{2}$$

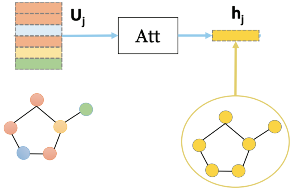
where $\boldsymbol{W_2} \in \mathbb{R}^{D\times D}$ is a learnable weight matrix

## Stage II: Graph Embedding: Global Context-Aware Attention

- Based on $\boldsymbol{c}$, we can compute one attention weight for each node embedding $a_n$ by computing an inner product with $\boldsymbol{c}$
- Graph embedding $\boldsymbol{h} \in \mathbb{R}^D$ is the weighted sum of node embeddings:

$$\boldsymbol{h} = \sum_{n=1}^{N} f_2(\boldsymbol{u}_n^T \boldsymbol{c})\boldsymbol{u}_n \tag{3}$$

where $f_2(\cdot)$ is the **sigmoid function** $\sigma(\cdot)$, i.e. not normalized

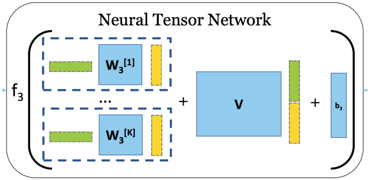


# Stage III: Graph-Graph Interaction: Neural Tensor Network (Strategy I)

- Instead of using inner product between graph embeddings, $\boldsymbol{h_i} \in \mathbb{R}^D$, $\boldsymbol{h_j} \in \mathbb{R}^D$ we use Neural Tensor Networks to model the relations:

$$g(\boldsymbol{h_i}, \boldsymbol{h_j}) = sigma(\boldsymbol{h_i^T} \boldsymbol{W_3^{[1:K]}} \boldsymbol{h_j} + \boldsymbol{V} \begin{bmatrix} \boldsymbol{h_i} \\ \boldsymbol{h_j} \end{bmatrix}) \tag{4}$$

where $\boldsymbol{W_3^{[1:K]}} \in \mathbb{R}^{D \times D \times K}$ is a weight tensor, [] denotes the concatenation operation, $\boldsymbol{V} \in \mathbb{R}^{K \times 2D}$ is a weight vector, $K$ is a hyperparameter controlling the number of interaction (similarity) scores for each graph embedding pair



## Stage IV: Graph Similarity Score Computation

- MLP used to output final similarity score from neural tensor vector.
- $\hat{s_{ij}} \in \mathbb{R}$ is predicted and compared against the ground-truth similarity score $s(\mathcal{G}_i, \mathcal{G}_j))$:

$$\mathcal{L} = \frac{1}{|\mathcal{D}|} \sum_{(i,j) \in \mathcal{D}} (\hat{s_{ij}} - s(\mathcal{G}_i, \mathcal{G}_j))^2 \tag{5}$$

where $\mathcal{D}$ is the set of training graph pairs, and $s(\mathcal{G}_i, \mathcal{G}_j)$ is the ground-truth similarity between $\mathcal{G}_i$ and $\mathcal{G}_j$.

## Limitations of Strategy One

- The node-level information such as the node feature distribution and graph size may be lost by the graph-level embedding
- In many cases, differences between two graphs lie in small substructures and can't be reflected by a graph level embedding
- To overcome these limitations, consider using the node-level embeddings directly

## Stage III: Graph-Graph Interaction: Pairwise Node Comparison (Strategy II)

- If $\mathcal{G}_i$ has $N_i$ nodes and $\mathcal{G}_j$ has $N_j$ nodes, there would be $N_i N_j$ pairwise interaction scores, obtained by $\boldsymbol{S} = \sigma(\boldsymbol{U_i U_j^T})$, where $\boldsymbol{U_i} \in \mathbb{R}^{N_i \times D}$ and $\boldsymbol{U_j} \in \mathbb{R}^{N_j \times D}$ are the node embeddings of $\mathcal{G}_i$ and $\mathcal{G}_j$, respectively.

## Stage III: Graph-Graph Interaction: Pairwise Node Comparison (Strategy II)

- If $\mathcal{G}_i$ has $N_i$ nodes and $\mathcal{G}_j$ has $N_j$ nodes, there would be $N_i N_j$ pairwise interaction scores, obtained by $\boldsymbol{S} = \sigma(\boldsymbol{U_i} \boldsymbol{U_j^T})$, where $\boldsymbol{U_i} \in \mathbb{R}^{N_i \times D}$ and $\boldsymbol{U_j} \in \mathbb{R}^{N_j \times D}$ are the node embeddings of $\mathcal{G}_i$ and $\mathcal{G}_j$, respectively.
- One simple way to utilize $\boldsymbol{S}$ is to vectorize it: $\mathrm{vec}(\boldsymbol{S}) \in \mathbb{R}^{N_i N_j}$, and feed to the fully connected layers. However, there is usually no natural ordering between graph nodes

## Stage III: Graph-Graph Interaction: Pairwise Node Comparison (Strategy II)

- To ensure the model is invariant to the graph representations, we extract its histogram features: $\mathrm{hist}(\boldsymbol{S}) \in \mathbb{R}^B$, where $B$ is a hyperparameter that controls the number of bins in the histogram.
- The histogram feature vector is normalized and concatenated with the graph-level interaction scores $g(\boldsymbol{h_i}, \boldsymbol{h_j})$, and fed to the MLP

## Outline

## Datasets

| Dataset | Graph Meaning | #Graphs | #Pairs |
|---|---|---|---|
| **AIDS** | Chemical Compounds | 700 | 490K |
| **LINUX** | Program Dependency Graphs | 1000 | 1M |
| **IMDB** | Actor/Actress Ego-Networks | 1500 | 2.25M |

## AIDS Chemical Compounds

| Method | mse($10^{-3}$) | $\rho$ | $\tau$ | p@10 | p@20 |
|---|---|---|---|---|---|
| **Beam** | 12.090 | 0.609 | 0.463 | **0.481** | 0.493 |
| **Hungarian** | 25.296 | 0.510 | 0.378 | 0.360 | 0.392 |
| **VJ** | 29.157 | 0.517 | 0.383 | 0.310 | 0.345 |
| **SimpleMean** | 3.115 | 0.633 | 0.480 | 0.269 | 0.279 |
| **HierarchicalMean** | 3.046 | 0.681 | 0.629 | 0.246 | 0.340 |
| **HierarchicalMax** | 3.396 | 0.655 | 0.505 | 0.222 | 0.295 |
| **AttDegree** | 3.338 | 0.628 | 0.478 | 0.209 | 0.279 |
| **AttGlobalContext** | 1.472 | 0.813 | 0.653 | 0.376 | 0.473 |
| **AttLearnableGC** | 1.340 | 0.825 | 0.667 | 0.400 | 0.488 |
| **SimGNN** | **1.189** | **0.843** | **0.690** | **0.421** | **0.514** |

# Program Dependence Graphs generated from the Linux kernel

| Method | mse($10^{-3}$) | $\rho$ | $\tau$ | p@10 | p@20 |
|---|---|---|---|---|---|
| **Beam** | 12.090 | 0.609 | 0.463 | **0.481** | 0.493 |
| **Hungarian** | 25.296 | 0.510 | 0.378 | 0.360 | 0.392 |
| **VJ** | 29.157 | 0.517 | 0.383 | 0.310 | 0.345 |
| **SimpleMean** | 3.115 | 0.633 | 0.480 | 0.269 | 0.279 |
| **HierarchicalMean** | 3.046 | 0.681 | 0.629 | 0.246 | 0.340 |
| **HierarchicalMax** | 3.396 | 0.655 | 0.505 | 0.222 | 0.295 |
| **AttDegree** | 3.338 | 0.628 | 0.478 | 0.209 | 0.279 |
| **AttGlobalContext** | 1.472 | 0.813 | 0.653 | 0.376 | 0.473 |
| **AttLearnableGC** | 1.340 | 0.825 | 0.667 | 0.400 | 0.488 |
| **SimGNN** | **1.189** | **0.843** | **0.690** | **0.421** | **0.514** |

## Actor/Actress Ego-Networks

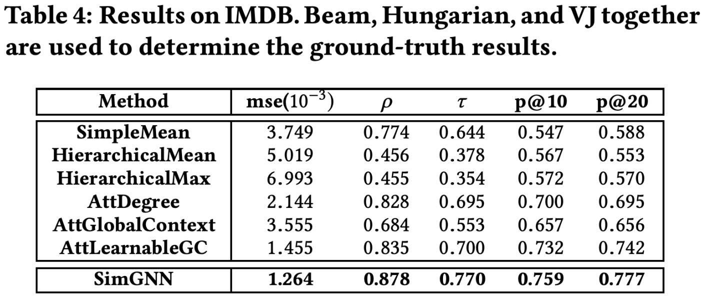


**Table 4: Results on IMDB. Beam, Hungarian, and VJ together are used to determine the ground-truth results.**

| Method | mse($10^{-3}$) | $\rho$ | $\tau$ | p@10 | p@20 |
|---|---|---|---|---|---|
| **SimpleMean** | 3.749 | 0.774 | 0.644 | 0.547 | 0.588 |
| **HierarchicalMean** | 5.019 | 0.456 | 0.378 | 0.567 | 0.553 |
| **HierarchicalMax** | 6.993 | 0.455 | 0.354 | 0.572 | 0.570 |
| **AttDegree** | 2.144 | 0.828 | 0.695 | 0.700 | 0.695 |
| **AttGlobalContext** | 3.555 | 0.684 | 0.553 | 0.657 | 0.656 |
| **AttLearnableGC** | 1.455 | 0.835 | 0.700 | 0.732 | 0.742 |
| **SimGNN** | **1.264** | **0.878** | **0.770** | **0.759** | **0.777** |

## Outline

## Conclusion

- Used GNN for embedding entire graphs to do graph similarity matching (e.g. querying similary graphs)
- Proposed method is empirically good, but I think the graph embedding and edge similarity parts could be improved
- No edge features used