CEGS 中集成边角色信息的 GNN 策略评估

April 23, 2025

1 引言

下面的所有策略均在通用的 GNN Message Passing 框架下运作[2]。该框架通过迭代地聚合邻居信息来更新中心节点的表示。边特征 x_{uv} (在 PyTorch Geometric 等库中常称为 `edge_attr`) 可以在消息构建、聚合或节点更新步骤中被整合[7]。评估这些策略的关键在于理解边角色向量 x_{uv} 如何影响从邻居 u 传递到中心节点 v 的消息 m_{uv} 或其聚合过程。

2 GNN 集成策略

2.1 策略一：修改 GraphSAGE 模型

CEGS 当前使用了 GraphSAGE 模型。标准 GraphSAGE 通常仅聚合邻居节点特征[2]，本策略旨在对其进行修改以融入边信息 x_{uv}。

2.1.1 方法 1.1: 边条件聚合

- 核心思想: 在 GraphSAGE 的消息构建（Message Calculation）阶段，让生成的消息不仅依赖于邻居节点 u 的表示 h_u^{l-1}，也显式地依赖于连接边 e_{uv} 的角色特征 x_{uv}[8]。
- 具体实现: 一种常见且有效的方式是将邻居节点特征 h_u^{l-1} 和边特征 x_{uv} 进行拼接（CONCAT），然后通过一个可学习的线性层（权重矩阵为 W_M^l）和激活函数（如 ReLU）来生成边感知的消息向量 m_{uv}^l。

$$m_{uv}^l = \text{Activation}(W_M^l \cdot \text{CONCAT}(h_u^{l-1}, x_{uv}))$$

  - h_u^{l-1}: 邻居节点 u 在第 $l-1$ 层的嵌入向量（状态向量）。
  - x_{uv}: 连接节点 u 和 v 的边 e_{uv} 的角色嵌入向量。
  - CONCAT$(\cdot,\cdot)$: 向量拼接操作，将节点信息和边信息合并。
  - W_M^l: 第 l 层消息计算步骤的可学习权重矩阵。其维度需要匹配拼接后向量的维度。
  - Activation$(\cdot)$: 非线性激活函数，例如 ReLU。
  - m_{uv}^l: 从邻居 u 传递给 v 的、包含了边角色信息的消息向量。
- 聚合与更新: 生成消息后，使用标准的 GraphSAGE 聚合器（如 Mean, Max, LSTM 等）聚合来自所有邻居的消息，并结合中心节点上一层的表示 h_v^{l-1} 来更新其当前层表示 h_v^l。

$$h_{\mathcal{N}(v)}^l \leftarrow \text{AGGREGATE}_l(\{m_{uv}^l, \forall u \in \mathcal{N}(v)\})$$

$$h_v^l \leftarrow \sigma(W^l \cdot \text{CONCAT}(h_v^{l-1}, h_{\mathcal{N}(v)}^l))$$

- $h^l_{\mathcal{N}(v)}$: 节点 v 在第 l 层聚合到的邻域表示。
- AGGREGATE$_l$: 第 l 层的聚合函数。
- h^l_v: 节点 v 在第 l 层的最终嵌入向量。
- W^l: 第 l 层更新步骤的可学习权重矩阵。
- σ: 更新步骤的非线性激活函数。

- 观点: 此方法紧密耦合了节点和边的信息[8]。聚合到节点 v 的信息显式地受到其入边角色的调节，使其对局部连接模式更敏感。但代价是模型复杂度和参数量（主要在 W^l_M）相较于标准 GraphSAGE 有所增加。这种思想与边条件卷积 (Edge Conditioned Convolution, ECC) 类似[8]。

2.1.2 方法 1.2: 基于注意力的聚合

- 核心思想: 借鉴图注意力网络 (GAT)[8] 或其改进版 GATv2[6] 的思想，在聚合邻居信息时，使用注意力机制动态计算邻居的权重。该权重 α_{uv} 的计算应同时依赖于中心节点 v 的表示 h^{l-1}_v、邻居节点 u 的表示 h^{l-1}_u 以及边 e_{uv} 的特征 x_{uv}。

- 具体实现:

1. 计算注意力系数: 首先计算一个原始的、未归一化的注意力分数 e_{uv}。这通常通过一个注意力机制函数 a 实现，该函数可以接收经过变换（通过可学习矩阵 W_Q, W_K, W_E）的节点和边表示。一种常见的形式（受 GAT 启发）是:

$$e_{uv} = a(W_Q h^{l-1}_v, W_K h^{l-1}_u, W_E x_{uv})$$

   - $W_Q h^{l-1}_v, W_K h^{l-1}_u, W_E x_{uv}$: 分别代表中心节点的需求，邻居节点提供的信息和连接关系的附加信息，类比 QKV。
   - $a(\cdot)$: 注意力机制函数。这里有多种实线方式：将三个输入拼接后通过一个 MLP；或者计算 q,k 的点积得分，并用 e 进行某种形式的修正；或者如 GAT-Edge 那样，将三者拼接后通过一个线性层和激活函数。

2. 归一化注意力权重: 使用 Softmax 函数对节点 v 的所有邻居 $u \in \mathcal{N}(v)$ 的注意力分数进行归一化，得到最终的注意力权重 α_{uv}。

$$\alpha_{uv} = \text{softmax}_u(e_{uv}) = \frac{\exp(\text{LeakyReLU}(e_{uv}))}{\sum_{k\in\mathcal{N}(v)} \exp(\text{LeakyReLU}(e_{kv}))}$$

3. 加权聚合邻居信息: 使用计算得到的注意力权重 α_{uv} 对经过变换（通过可学习矩阵 W_V）的邻居节点表示进行加权求和。

$$h^l_{\mathcal{N}(v)} \leftarrow \sum_{u\in\mathcal{N}(v)} \alpha_{uv}(W_V h^{l-1}_u)$$

   - W_V: 对邻居节点表示进行变换的可学习值矩阵。
   - $h^l_{\mathcal{N}(v)}$: 注意力加权聚合后的邻域表示向量。

4. 更新节点表示: 与 GraphSAGE 类似，使用 $h^l_{\mathcal{N}(v)}$ 和 h^{l-1}_v 计算 h^l_v，通常是拼接后通过一个线性层和激活函数。

$$h^l_v \leftarrow \sigma(W^l \cdot \text{CONCAT}(h^{l-1}_v, h^l_{\mathcal{N}(v)}))$$

- 观点: 注意力机制提供了一种比方法 1.1 更灵活的方式来整合边角色[8]。模型可以学习到更复杂的、上下文相关的依赖关系。优先考虑 GATv2 架构是合理的，因为它解决了 GAT 的" 静态注意力" 问题，提供了更强的表达能力[6]。但此方法比方法 1.1 更复杂，注意力计算也会带来额外的计算开销。

2.2 策略二：替换为原生支持边属性的 GNN 模型

直接采用设计上就考虑了边属性或多关系图的 GNN 模型。

2.2.1 方法 2.1: R-GCN (Relational Graph Convolutional Network)

- 核心思想: 将每个边角色视为一个独立的关系类型 $r \in R$。模型为每种关系类型 r 学习一个专属的变换矩阵 $W_r^{(l)}$，并分别对来自不同关系类型的邻居信息进行变换和聚合[1]。
- 具体实现: R-GCN 的核心更新规则大致如下[1]:

$$h_v^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{u\in\mathcal{N}_r(v)}\frac{1}{c_{v,r}}W_r^{(l)}h_u^{(l)} + W_0^{(l)}h_v^{(l)}\right)$$

  - $h_v^{(l)}$: 节点 v 在第 l 层的嵌入向量。
  - R: 图中所有关系类型（边角色）的集合。
  - $\mathcal{N}_r(v)$: 节点 v 在关系 r 下的邻居节点集合。
  - $W_r^{(l)}$: 与关系类型 r 相关的、在第 l 层的可学习权重矩阵。这是 R-GCN 的关键[1]。
  - $W_0^{(l)}$: 用于变换节点自身表示（自环）的权重矩阵[1]。
  - $c_{v,r}$: 归一化常数，通常取 $c_{v,r} = |\mathcal{N}_r(v)|$[1]。
  - σ: 非线性激活函数。
- 观点: [9] 论文明确指出，对于处理离散、类别化的边角色，R-GCN 提供了最直接且理论上最匹配的方法。它明确地区分了不同连接类型的影响。主要挑战在于，当关系类型（边角色）数量 $|R|$ 很大时，参数数量会线性增长，可能需要采用基分解或块对角分解等正则化技术来减少参数量[4]。另外，是否采用多任务学习?

2.2.2 方法 2.2: GATv2 (作为独立模型)

- 核心思想: 不再修改现有的 GraphSAGE，而是直接采用 Graph Attention Network v2 (GATv2) 这一更先进的图注意力网络架构作为 GNN 模型。GATv2 的主要优势在于其动态注意力机制，能够捕捉更复杂的节点间依赖关系。同时，我们将编码后的边角色嵌入向量 x_{uv} 作为 GATv2 层的边属性 (`edge_attr`) 输入，让模型在计算注意力时能够利用边的信息。
- **GATv2** 注意力机制:
  - GATv2 与原始 GAT 的核心区别在于计算注意力系数 e_{ij} 的方式。GATv2 旨在解决 GAT 的"静态注意力"问题（即注意力排序可能与查询节点无关），使得注意力计算更能响应查询节点的特征[5]。
  - GATv2 计算注意力系数 e_{ij} 的基础公式为:

$$e_{ij} = \mathbf{a}^T \text{LeakyReLU}(\mathbf{W}[\mathbf{h}_i || \mathbf{h}_j])$$

    - $\mathbf{h}_i, \mathbf{h}_j$: 分别是中心节点 i 和邻居节点 j 在当前层的特征向量。
    - $||$: 表示向量拼接操作。GATv2 先拼接节点特征。
    - $\mathbf{W}$: 一个共享的可学习线性变换（权重矩阵），应用于拼接后的向量。这是与 GAT 的关键不同，GAT 是先分别变换 $\mathbf{h}_i$ 和 $\mathbf{h}_j$ 再拼接。
    - LeakyReLU$(\cdot)$: LeakyReLU 激活函数。

* $\mathbf{a}^T$: 一个可学习的注意力头（权重向量），用于将变换后的高维向量映射为一个标量分数。

– 这个公式使得注意力系数 e_{ij} 的计算同时依赖于 h_i 和 h_j 在变换 $\mathbf{W}$ 之后的结果，从而实现了更动态、表达能力更强的注意力[5]。

- 整合边特征 **(x_{uv} 作为 `edge_attr`)**:

– 原始的 GATv2 公式没有显式包含边特征。但在实际应用中（例如使用 PyTorch Geometric 库中的 `GATv2Conv` 层），通常支持传入 `edge_attr` 参数[9]。

– 将边特征 x_{uv} 融入 GATv2 注意力计算的具体实现方式有多种，但核心思想是将边的信息也加入到计算注意力分数 e_{ij} 的过程中。一种常见且合理的方式（类似于 GAT-Edge 或我们之前讨论的通用注意力形式）是：

* 修改注意力系数计算: 将经过变换的边特征也纳入考虑。例如，可以想象将边特征 x_{ij} 通过一个专门的变换矩阵 $\mathbf{W}_E$ 变换后，与拼接并变换后的节点特征一起输入到最终的注意力头 $\mathbf{a}^T$ 之前。一个可能的（非标准 GATv2 公式，但概念上可行）形式是：

$$e_{ij} = \mathbf{a}^T \text{LeakyReLU}(\mathbf{W}[\mathbf{h}_i || \mathbf{h}_j] + \mathbf{W}_E \mathbf{x}_{ij})$$

或者，更常见的可能是将变换后的边特征也进行拼接（这更接近 GAT-Edge 的做法）：

$$e_{ij} = \mathbf{a}^T \text{LeakyReLU}(\mathbf{W}'[(\mathbf{W}[\mathbf{h}_i || \mathbf{h}_j]) || (\mathbf{W}_E \mathbf{x}_{ij})])$$

这里的 $\mathbf{W}_E$ 是用于变换边特征的可学习矩阵。$\mathbf{W}'$ 和 $\mathbf{a}^T$ 需要调整维度以适应加入边特征后的拼接向量。

– 关键点: 无论具体实现细节如何，目标都是让计算出的注意力分数 e_{ij} 同时反映节点 i, j 的特征以及连接边 e_{ij} 的特征 x_{ij}。

- 注意力权重计算与聚合:

– 得到包含边信息的注意力系数 e_{ij} 后，后续步骤与标准 GAT/GATv2 相同：

* 归一化: 使用 Softmax 函数将 e_{ij} 归一化，得到最终的注意力权重 α_{ij}:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

* 加权聚合: 使用 α_{ij} 对邻居节点 j 经过值变换 **($\mathbf{W}_V$)** 后的特征进行加权求和，得到节点 i 的下一层表示 $\mathbf{h}'_i$:

$$\mathbf{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}_V \mathbf{h}_j\right)$$

· $\mathbf{W}_V$ 是另一个可学习的线性变换矩阵。

· σ 是最终的激活函数（如 ELU）。

– GATv2 通常也使用多头注意力 **(Multi-head Attention)**，即将上述过程重复 K 次（每" 头" 使用独立的参数 $\mathbf{a}^T, \mathbf{W}, \mathbf{W}_E, \mathbf{W}_V$），然后将 K 个头的输出结果进行拼接或平均，以增强模型的稳定性和表达能力。

- 观点: 相比于 R-GCN 对每种边类型应用固定的变换矩阵 W_r，GATv2 计算的注意力权重 α_{ij} 是基于 h_i, h_j, x_{ij} 动态生成的，更能捕捉特定场景下的上下文信息。当边的关系类型（角色）数量非常多时，标准 R-GCN 的参数量会线性增长。而 GATv2 的主要参数（如 $\mathbf{W}, \mathbf{a}^T, \mathbf{W}_V, \mathbf{W}_E$）是在所有节点和边之间共享的（除了多头机制会复制参数），因此在这种情况下 GATv2 可能更节省参数[9]。

但是 R-GCN 中学习到的 W_r 可以相对直接地反映不同关系类型的作用。而 GATv2 的注意力权重 α_{ij} 虽然也提供了边的重要性信息，但解释这些权重是如何具体区分不同的原始边角色或它们是如何综合节点与边信息得到的，通常更复杂一些。

2.2.3 方法 2.3: 其他异构图神经网络 (HGT, HAN 等)

- 核心思想: 采用专门为异构信息网络（Heterogeneous Information Networks, HINs）设计的更复杂的 GNN 架构，如 HGT (Heterogeneous Graph Transformer) 或 HAN (Heterogeneous Graph Attention Network)。这些模型通常使用元路径（meta-paths）或异构注意力机制来处理不同类型的节点和边[10]。
- 观点: 就我们的任务而言，图中主要的异构性仅体现在边类型上，这些模型可能显得过于复杂（"overkill"）[9]。它们通常需要定义元路径或复杂的类型映射，实现复杂度较高。但如果网络意图分析表明存在重要的、涉及特定边角色序列的多跳关系模式，这些模型可能提供更强的建模能力[9]。

3 总结与对比

下表总结了上述 GNN 策略在集成类别化边角色时的关键特性：

Table 1: 用于集成类别化边角色的 GNN 策略对比

策略	核心机制	优点	缺点	相关文献
修改 **GraphSAGE**				
1.1 边条件聚合	在消息计算中融合节点与边嵌入 ($f(h_u, x_{uv})$)	直接将边信息融入聚合；对局部连接敏感	增加模型复杂度和参数量；变换方式固定	[3]
1.2 注意力聚合 (GATv2)	注意力权重依赖节点和边嵌入 (α_{uv})	灵活，动态加权；能学习上下文重要性；GATv2 表达能力强	相较于 GraphSAGE 更复杂；注意力计算开销	[8]
替换模型				
2.1 R-GCN	每种边角色（关系）使用独立的变换矩阵 (W_r)	理论上最匹配离散角色；直接建模关系特异性	参数量随角色数量线性增长（需正则化）；变换方式固定	[1]
2.2 GATv2 (独立模型)	使用 edge_attr 的注意力机制	强大灵活；能捕捉复杂节点-边交互；参数效率可能优于 R-GCN（多角色时）	可能比 R-GCN 更难解释；注意力计算开销	[6]
2.3 其他异构 GNN	元路径、异构注意力等	能捕捉复杂的异构交互模式（多跳、多类型）	实现复杂；可能过度设计；需要定义元路径或完整异构模式	[10]

4 参考文献

References

[1] Modeling relational data with graph convolutional networks. *arXiv*, 2017.

[2] Graph neural networks. *arXiv*, 2018.

[3] Message passing attention networks for document understanding. *GitHub Pages*, 2020.

[4] Dgl tutorials: Relational graph convolutional network. *GitHub*, 2021.

[5] How attentive are graph attention networks? *ICLR*, 2022.

[6] Pytorch geometric documentation: Gatv2conv. *PyTorch Geometric*, 2022.

[7] Creating message passing networks —pytorch_geometric documentation. *PyTorch Geometric*, 2023.

[8] Gat-edge: Graph attention neural network with adjacent edge features. *OpenReview*, 2023.

[9] Gl-fusion: Rethinking the combination of graph neural network and large language model. *OpenReview*, 2024.

[10] Text classification model based on graph attention networks and adversarial training. *Applied Sciences*, 2024.