Network encoders with Transformers

GraphBERT and related models

Irina Nikishina, PhD
Universität Hamburg

some slides are borrowed from
WWW-18 Tutorial
“Representation Learning on Networks”
presented by Yure Leskovec

Course outline

1. The Transformer: motivation, original architecture and attention mechanism.
2. Transformer-based Encoders. Masked language models based on the Transformer architecture. BERT and related models.
3. Classification and sequence tagging with Transformers. Using encoders to generate feature representation for various NLU tasks.
4. Transformer-based Decoders. Generation of text based on the Transformer architecture. GPT and related decoders. Text generation methods. Prompt tuning.
5. Prompt and Instruction tuning. Reinforcement Learning from Human Feedback (RLHF), ChatGPT, and related models.
6. Sequence to sequence tasks: machine translation, text detoxification, question answering, dialogue. Technical tricks for training and inference: infrastructure and performance.
7. Multilingual language models based on the Transformer architecture.
8. Uncertainty estimation for Transformers and NLP
9. Efficient Transformers.
10. Compression of transformer models.
11. Network encoders with Transformers.
12. Multimodal and vision Transformers.
13. Transformers for tabular data.
14. Transformers for event sequences.

Graphs: recap

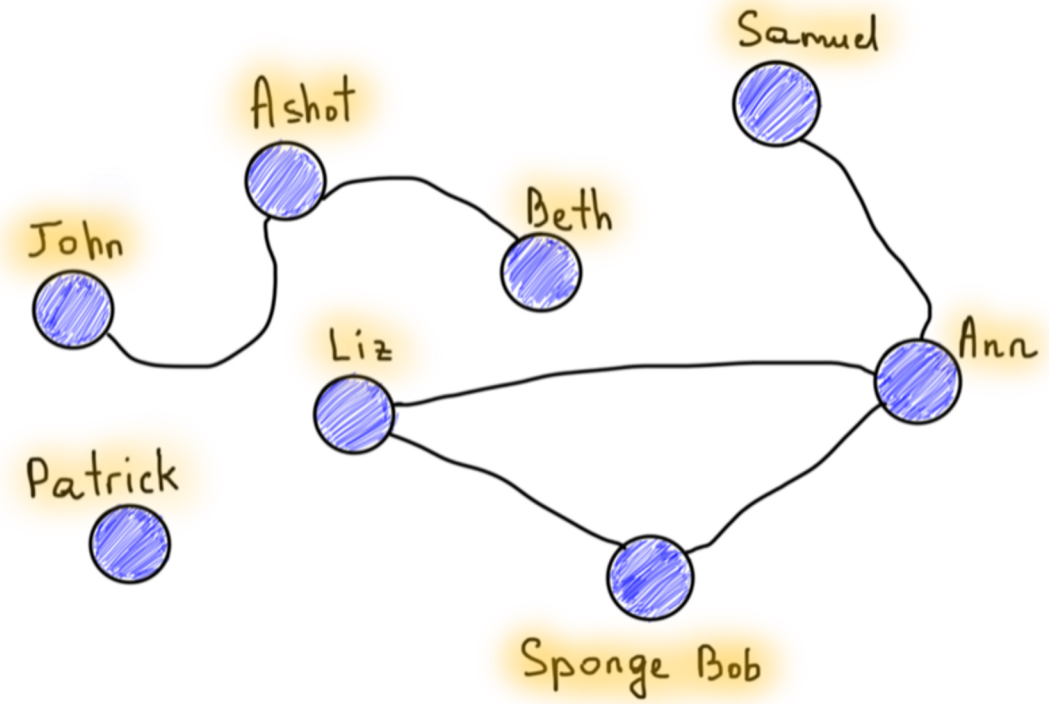
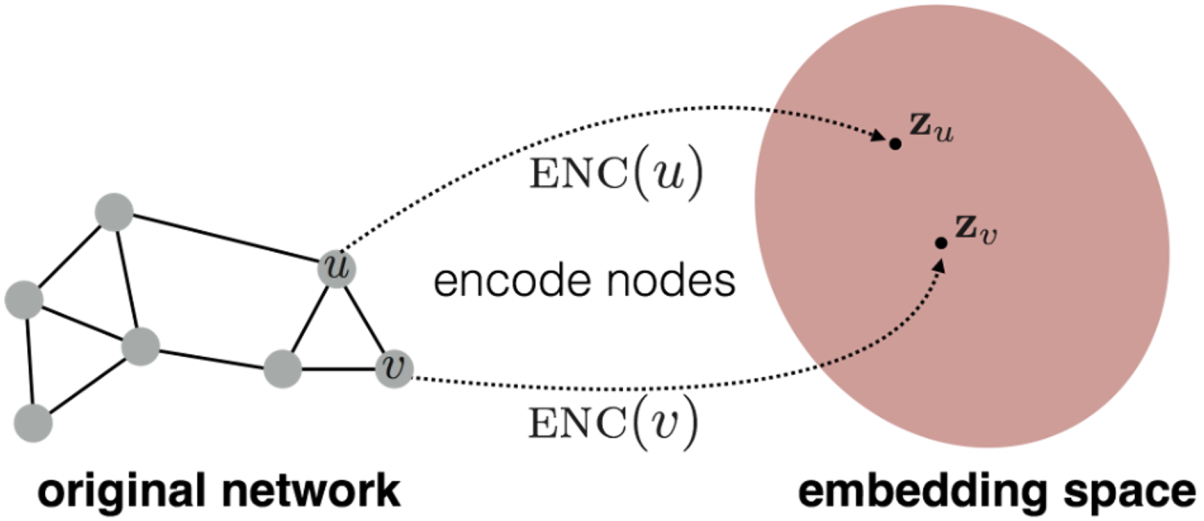



Схема метро Москвы 2023
www.mskmetro.de.ru
светло-серым цветом выделены станции, которые ремонтируются
Сокольническая
Замоскворецкая
Арбатско-Покровская
Филёвская
Кольцевая
Калужско-Рижская
Таганско-Краснопресненская
Калининско-Солнцевская
Серпуховско-Тимирязевская
Люблинско-Дмитровская
Бутовская (легкое метро)
Некрасовская
Московское центральное кольцо (МЦК)
Большая кольцевая линия (БКЛ)
Монорельсовая транспортная система
1 Красная площадь и Кремль
2 Храм Христа Спасителя
3 Поклонная гора
4 Третьяковская галерея
5 Останкинская башня
6 Усадьба Царицыно
7 Усадьба Коломенское
8 Зоопарк
9 Университет и Воробьевы горы
10 ВДНХ (ВВЦ)
11 Москва-Сити и Экспоцентр
12 Крокус Экспо

Node embeddings

Goal is to encode nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the original network.



Graph Neural Networks

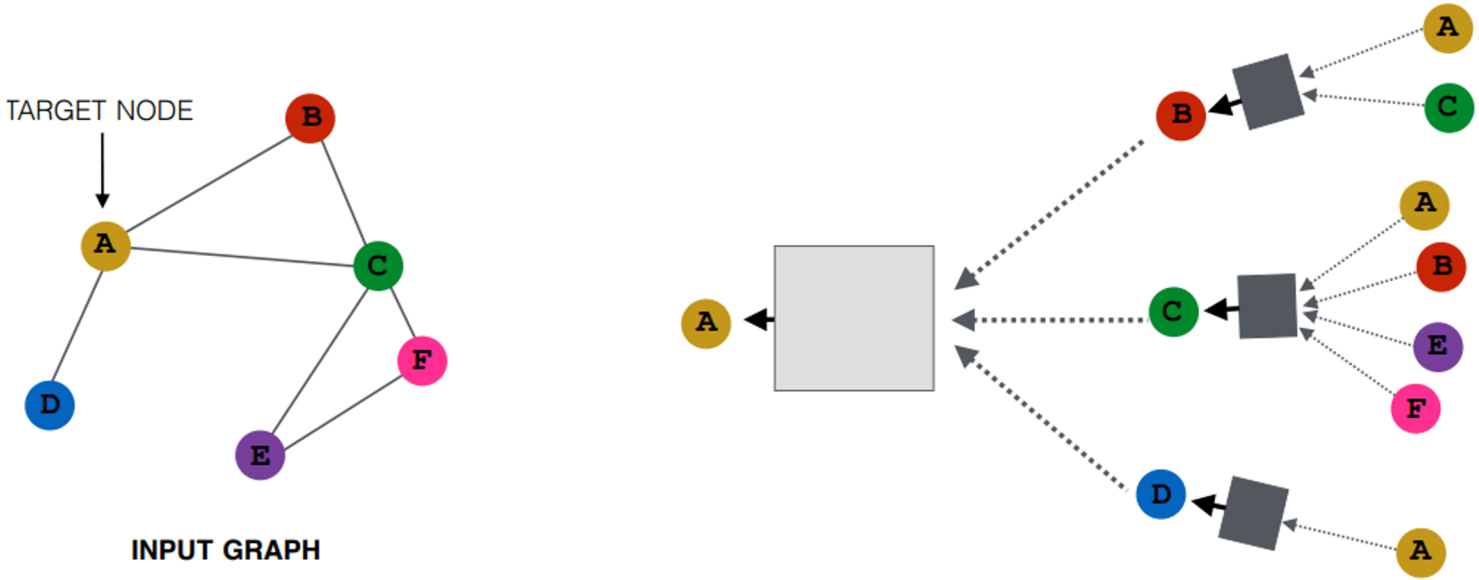
Assume we have a graph G:

- **V** is the vertex set.
- **A** is the adjacency matrix (assume binary).

$\mathbf{X} \in \mathbf{R}^{m \times |V|}$ is a matrix of node features.

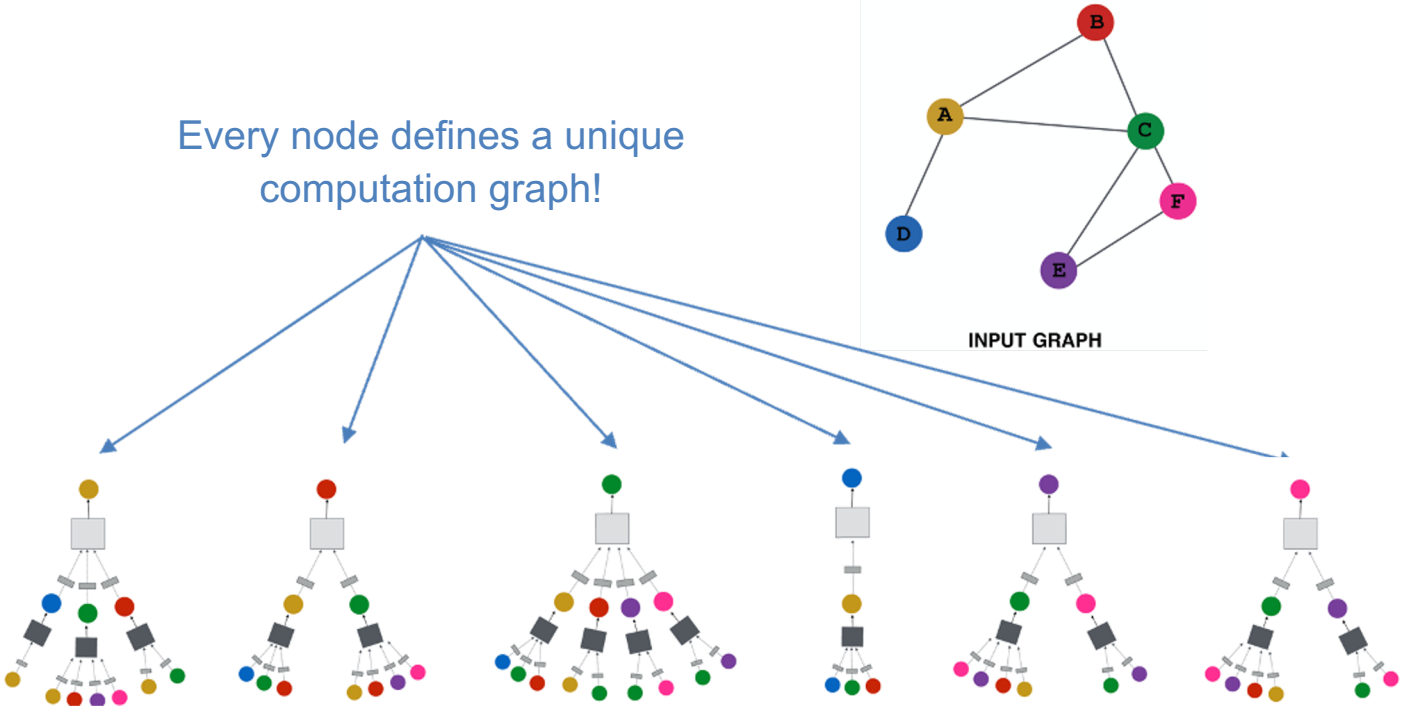
- Categorical attributes, text, image data
- E.g., profile information in a social network.
- Node degrees, clustering coefficients, etc.
- Indicator vectors (i.e., one-hot encoding of each node)

Graph Convolutional Networks



T.N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, *arXiv preprint* 31 *arXiv:1609.02907* (2016).

Graph Convolutional Networks



T.N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, *arXiv preprint* 31 *arXiv:1609.02907* (2016). https://arxiv.org/pdf/1609.02907.pdf

Graph Convolutional Networks

Trainable weight matrices
(i.e., what we learn)

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k \in \{1, ..., K\}$$

$$\mathbf{z}_v = \mathbf{h}_v^K$$

T.N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, *arXiv preprint* 31 *arXiv:1609.02907* (2016).

Graph Attention Networks (GAT)

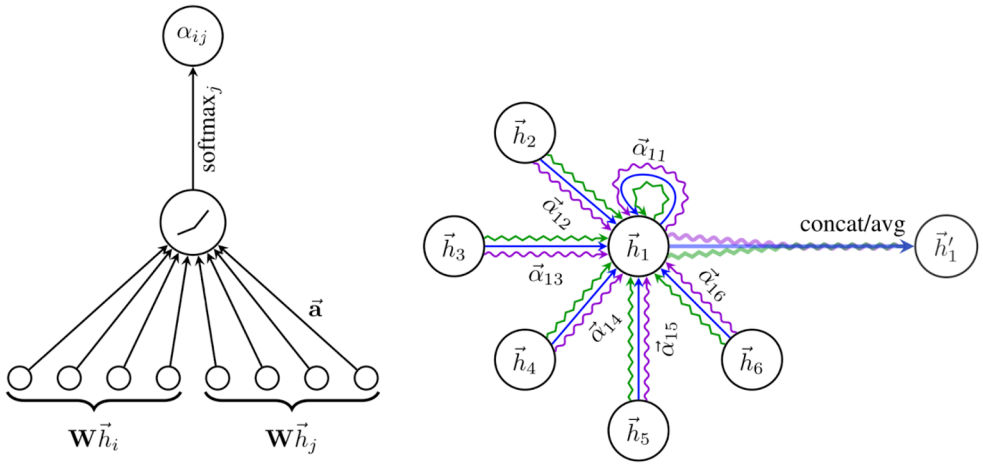
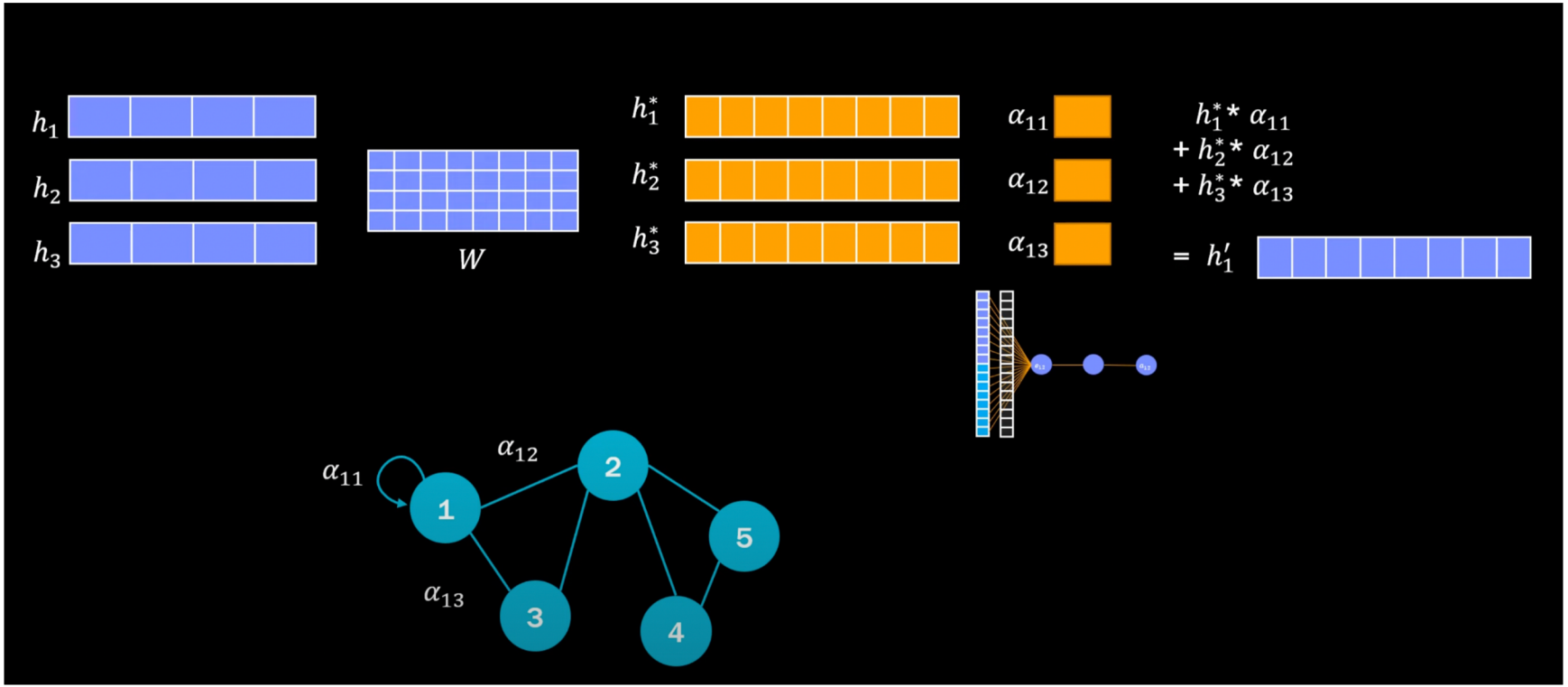


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, Graph Attention Networks, ICLR (2018). https://arxiv.org/pdf/1710.10903.pdf

Graph Attention Networks (GAT)



P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, Graph Attention Networks, ICLR (2018). https://arxiv.org/pdf/1710.10903.pdf

Graph-BERT

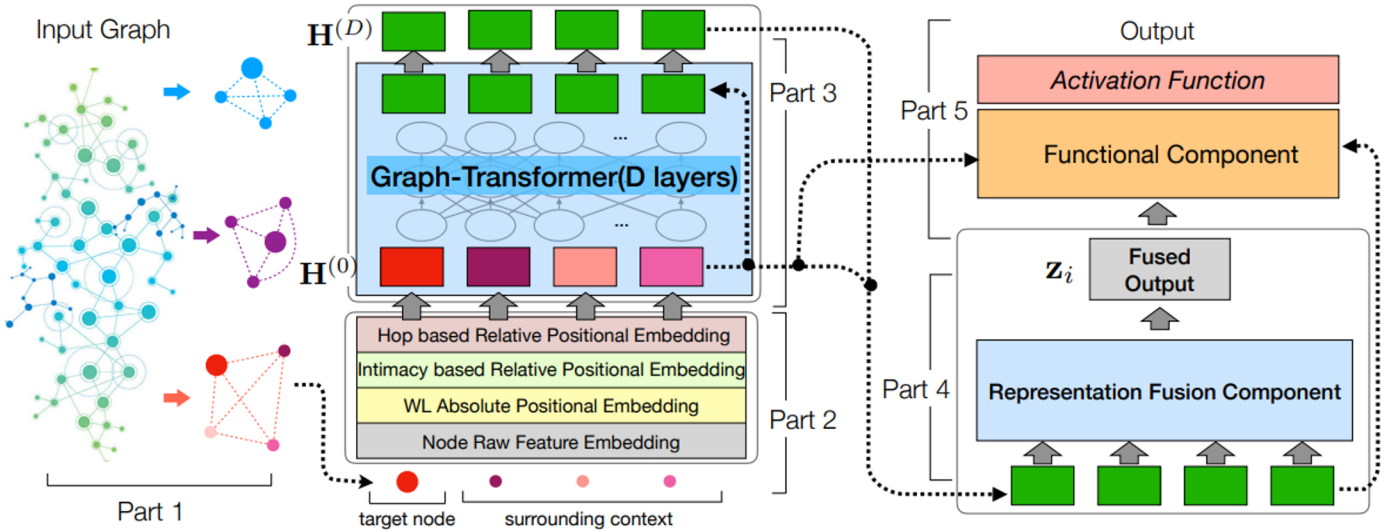


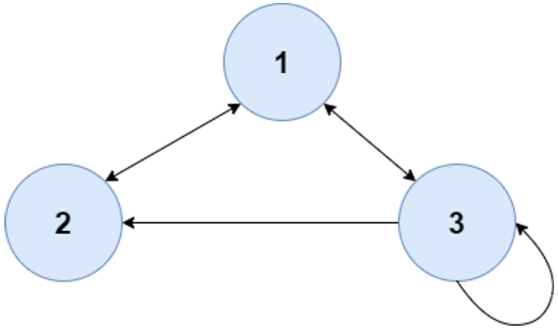
Figure 1: Architecture of the GRAPH-BERT Model. (Part 1: linkless subgraph batching; Part 2: node input vector embeddings; Part 3: graph transformer based encoder; Part 4: representation fusion; Part 5: functional component. Depending on the target application task, the function component will generate different output. In the sampled subgraphs, it covers both the target node and the surrounding context nodes.)

Zhang, Jiawei, et al. "*Graph-bert: Only attention is needed for learning graph representations.*" arXiv preprint arXiv:2001.05140 (2020).
https://arxiv.org/pdf/2001.05140.pdf

Subgraph sampling

$$\mathbf{S} = \alpha \cdot (\mathbf{I} - (1-\alpha) \cdot \bar{\mathbf{A}})^{-1}$$

$$\bar{\mathbf{A}} = \mathbf{A}\mathbf{D}^{-1}$$

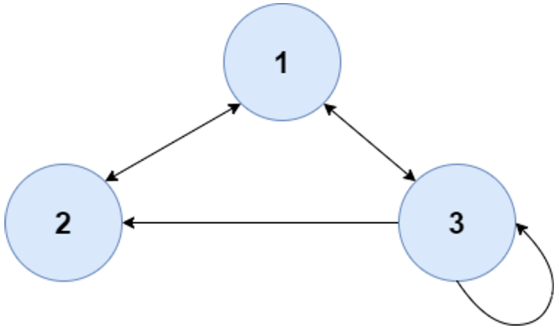


$$\mathbf{A} = \begin{vmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{vmatrix}$$

$$\mathbf{D} = \begin{vmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{vmatrix}$$

$$\bar{\mathbf{A}} = \begin{vmatrix} 0 & 1 & \frac{1}{3} \\ 0.5 & 0 & \frac{1}{3} \\ 0.5 & 0 & \frac{1}{3} \end{vmatrix}$$

Subgraph sampling



$$\mathbf{S} = \alpha \cdot (\mathbf{I} - (1-\alpha) \cdot \bar{\mathbf{A}})^{-1}$$

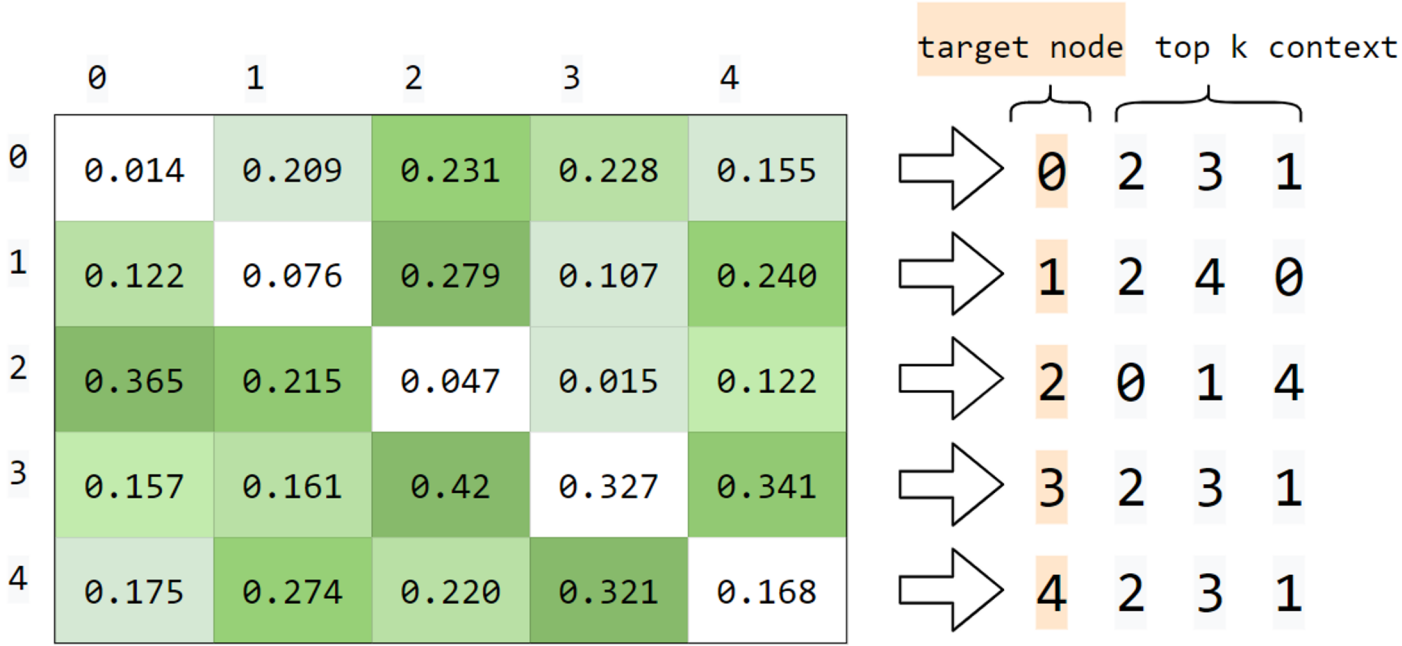
$$\bar{\mathbf{A}} = \mathbf{A}\mathbf{D}^{-1}$$

$$\mathbf{S} = (1-\alpha) \cdot \mathbf{A} + \alpha \cdot \frac{1}{n}\mathbf{I}$$

$$\mathbf{S} = (1-0.15) \cdot \begin{vmatrix} 0 & 1 & \frac{1}{3} \\ 0.5 & 0 & \frac{1}{3} \\ 0.5 & 0 & \frac{1}{3} \end{vmatrix} + 0.15 \cdot \begin{vmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{vmatrix}$$

$$\mathbf{S} = \begin{vmatrix} 0.04995 & 0.89995 & 0.333 \\ 0.47495 & 0.04995 & 0.333 \\ 0.47495 & 0.04995 & 0.333 \end{vmatrix}$$

Subgraph sampling



Positional embeddings

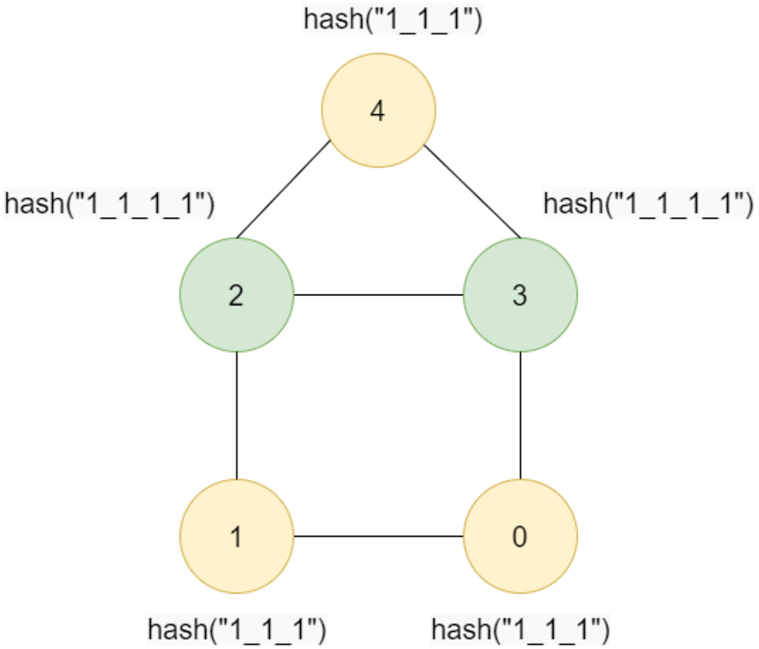
Weisfeiler-Lehman Absolute Role Embedding

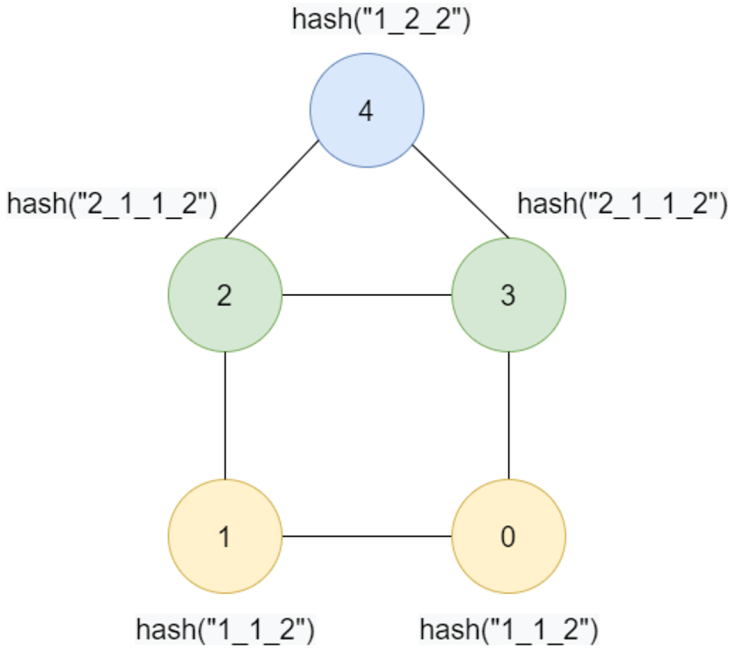
Intimacy based Relative Positional Embedding

Hop based Relative Distance Embedding

Positional embeddings

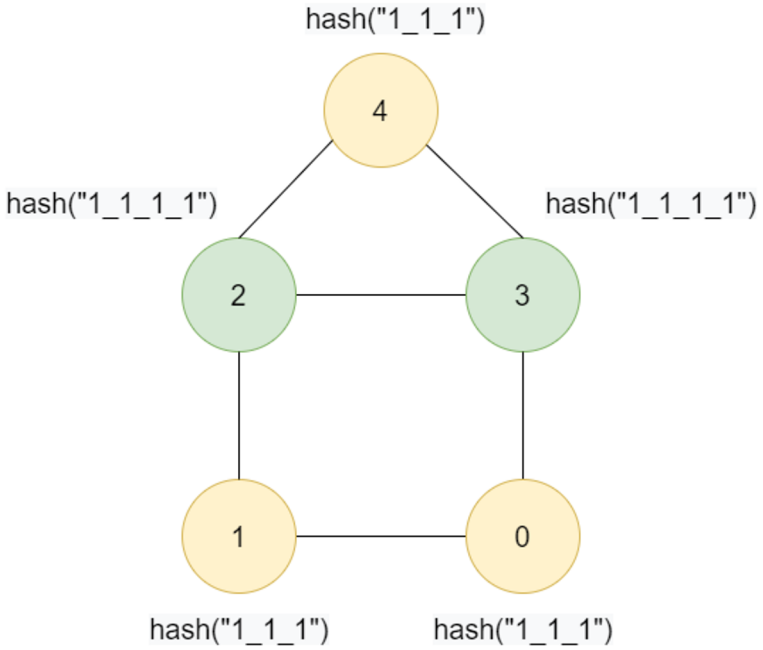
Weisfeiler-Lehman Absolute Role Embedding





Positional embeddings

Weisfeiler-Lehman Absolute Role Embedding

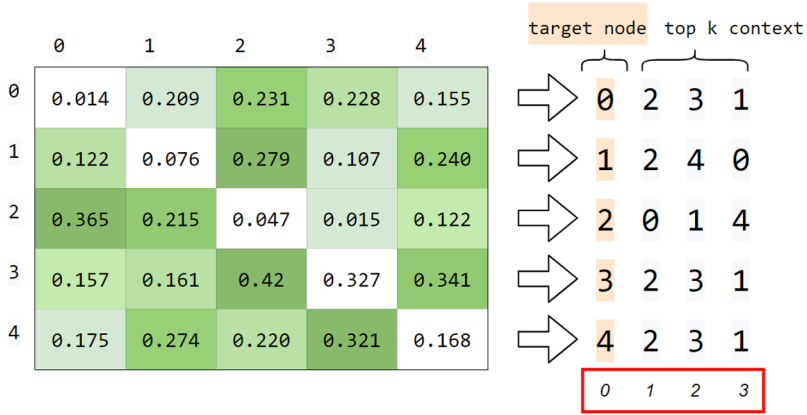


$$\mathbf{e}_j^{(r)} = \text{Position-Embed}\left(\text{WL}(v_j)\right)$$

$$= \left[sin\left(\frac{\text{WL}(v_j)}{10000^{\frac{2l}{d_h}}}\right), cos\left(\frac{\text{WL}(v_j)}{10000^{\frac{2l+1}{d_h}}}\right) \right]_{l=0}^{\lfloor \frac{d_h}{2} \rfloor}$$

Positional embeddings

Intimacy based Relative Positional Embedding



$$\mathbf{e}_j^{(p)} = \text{Position-Embed}\left(\mathrm{P}(v_j)\right) \in \mathbb{R}^{d_h \times 1}$$

Positional embeddings

Hop based Relative Distance Embedding

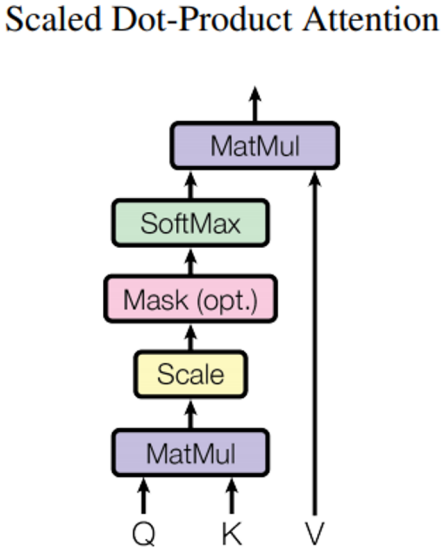
$$\mathbf{e}_j^{(d)} = \text{Position-Embed}\left(\mathrm{H}(v_j; v_i)\right) \in \mathbb{R}^{d_h \times 1}$$

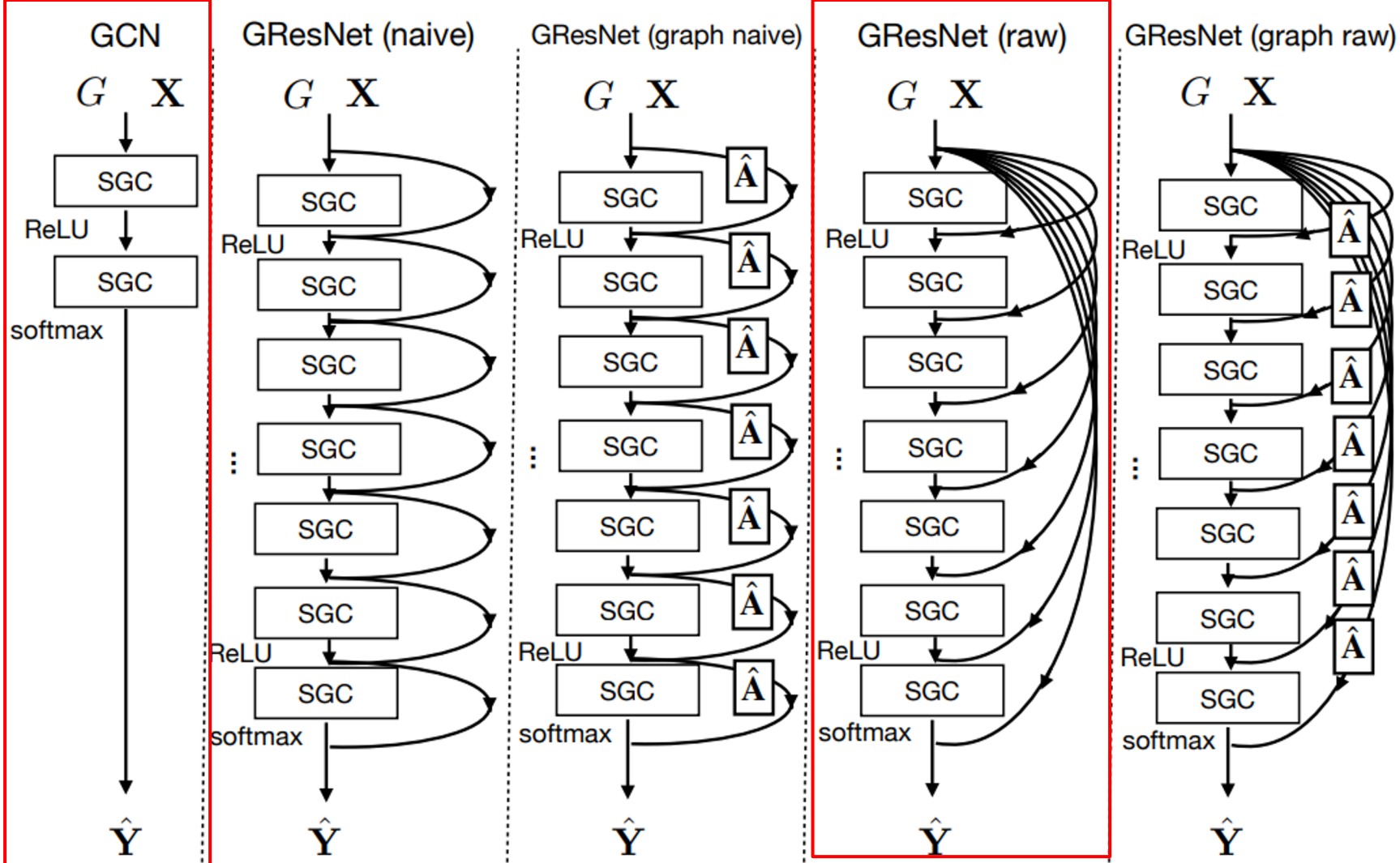
Graph Transformer based Encoder

$$\mathbf{H}^{(l)} = \text{G-Transformer}\left(\mathbf{H}^{(l-1)}\right)$$
$$= \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d_h}}\right)\mathbf{V} + \text{G-Res}\left(\mathbf{H}^{(l-1)}, \mathbf{X}_i\right)$$

where

$$\begin{cases} \mathbf{Q} &= \mathbf{H}^{(l-1)}\mathbf{W}_Q^{(l)}, \\ \mathbf{K} &= \mathbf{H}^{(l-1)}\mathbf{W}_K^{(l)}, \\ \mathbf{V} &= \mathbf{H}^{(l-1)}\mathbf{W}_V^{(l)}. \end{cases}$$




GCN
GResNet (naive)
GResNet (graph naive)
GResNet (raw)
GResNet (graph raw)
G X
SGC
ReLU
softmax
Â
Ŷ

Graph Transformer based Encoder

$$
\begin{cases}
\mathbf{H}^{(0)} = [\mathbf{h}_i^{(0)}, \mathbf{h}_{i,1}^{(0)}, \cdots, \mathbf{h}_{i,k}^{(0)}]^\top, \\
\mathbf{H}^{(l)} = \text{G-Transformer}\left(\mathbf{H}^{(l-1)}\right), \forall l \in \{1, 2, \cdots, D\}, \\
\mathbf{z}_i \quad = \text{Fusion}\left(\mathbf{H}^{(D)}\right).
\end{cases}
$$

GRAPH-BERT Learning

Task #1: Node Raw Attribute Reconstruction

$$\ell_1 = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \left\| \mathbf{x}_i - \hat{\mathbf{x}}_i \right\|_2$$

$$\hat{\mathbf{x}}_i = \mathrm{FC}(\mathbf{z}_i)$$

Task #2: Graph Structure Recovery

$$\ell_2 = \frac{1}{|\mathcal{V}|^2} \left\| \mathbf{S} - \hat{\mathbf{S}} \right\|_F^2$$

$\hat{\mathbf{S}} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ with entry $\hat{\mathbf{S}}(i,j) = \hat{s}_{i,j}$

Graph Transformer Networks

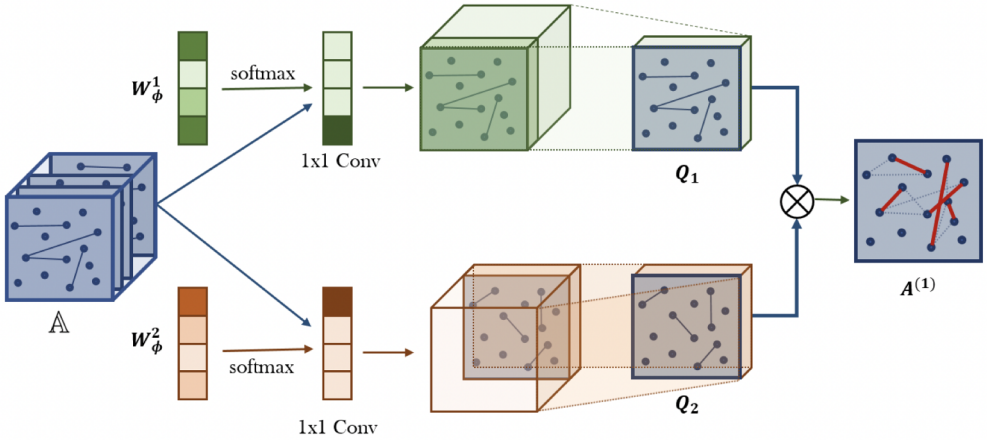


Figure 1: **Graph Transformer Layer** softly selects adjacency matrices (edge types) from the set of adjacency matrices $\mathbb{A}$ of a heterogeneous graph G and learns a new meta-path graph represented by $A^{(1)}$ via the matrix multiplication of two selected adjacency matrices Q_1 and Q_2. The soft adjacency matrix selection is a weighted sum of candidate adjacency matrices obtained by 1×1 convolution with non-negative weights from softmax(W_ϕ^1).

Yun, Seongjun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. "Graph transformer networks." *Advances in neural information processing systems* 32 (2019). https://arxiv.org/pdf/1911.06455.pdf

Graph Transformer Networks

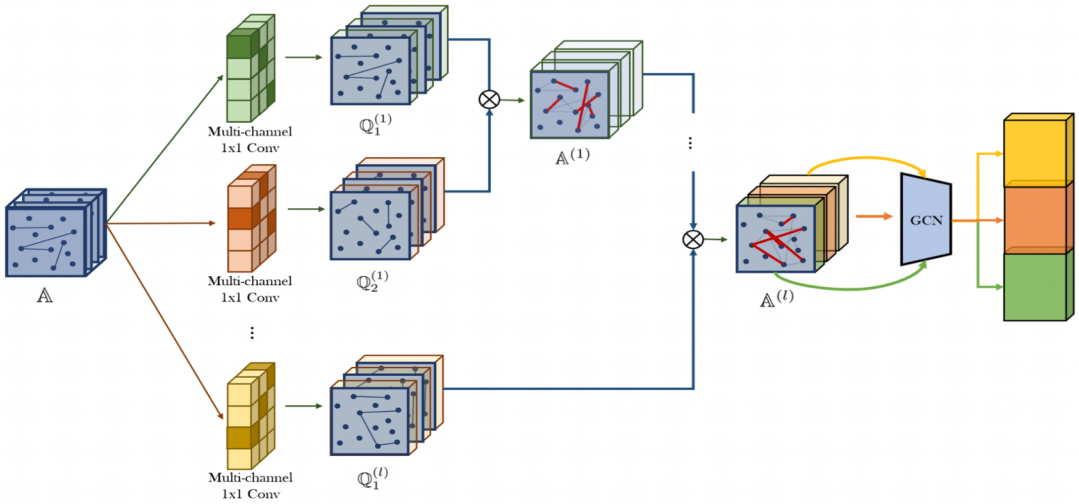


Figure 2: Graph Transformer Networks (GTNs) learn to generate a set of new meta-path adjacency matrices $\mathbb{A}^{(l)}$ using GT layers and perform graph convolution as in GCNs on the new graph structures. Multiple node representations from the same GCNs on multiple meta-path graphs are integrated by concatenation and improve the performance of node classification. $\mathbb{Q}_1^{(l)}$ and $\mathbb{Q}_2^{(l)} \in \mathbf{R}^{N \times N \times C}$ are intermediate adjacency tensors to compute meta-paths at the lth layer.

Yun, Seongjun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. "Graph transformer networks." *Advances in neural information processing systems* 32 (2019). https://arxiv.org/pdf/1911.06455.pdf

Heterogeneous Graph Transformer

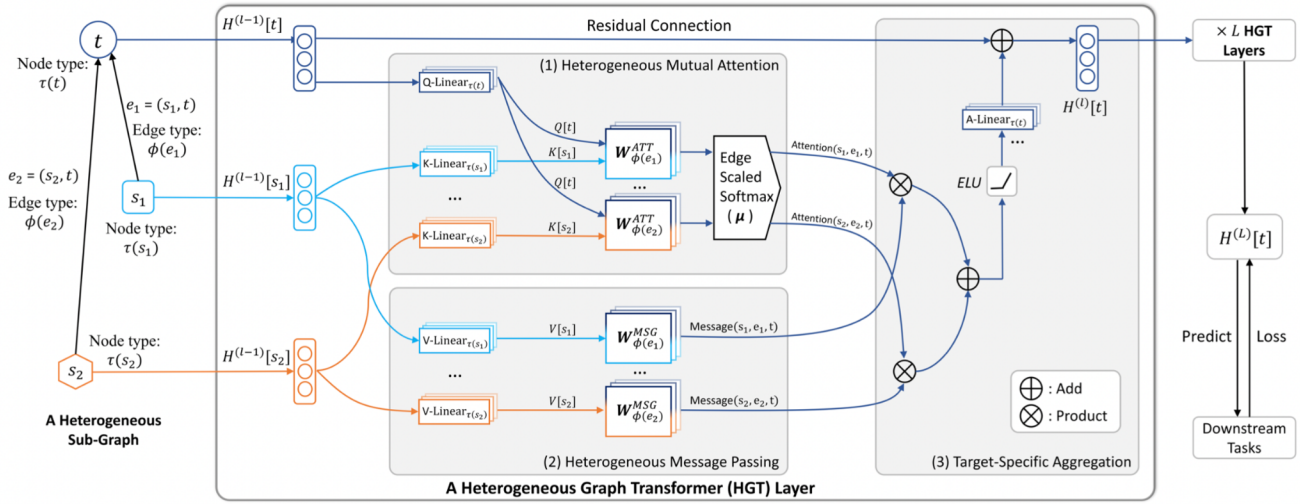
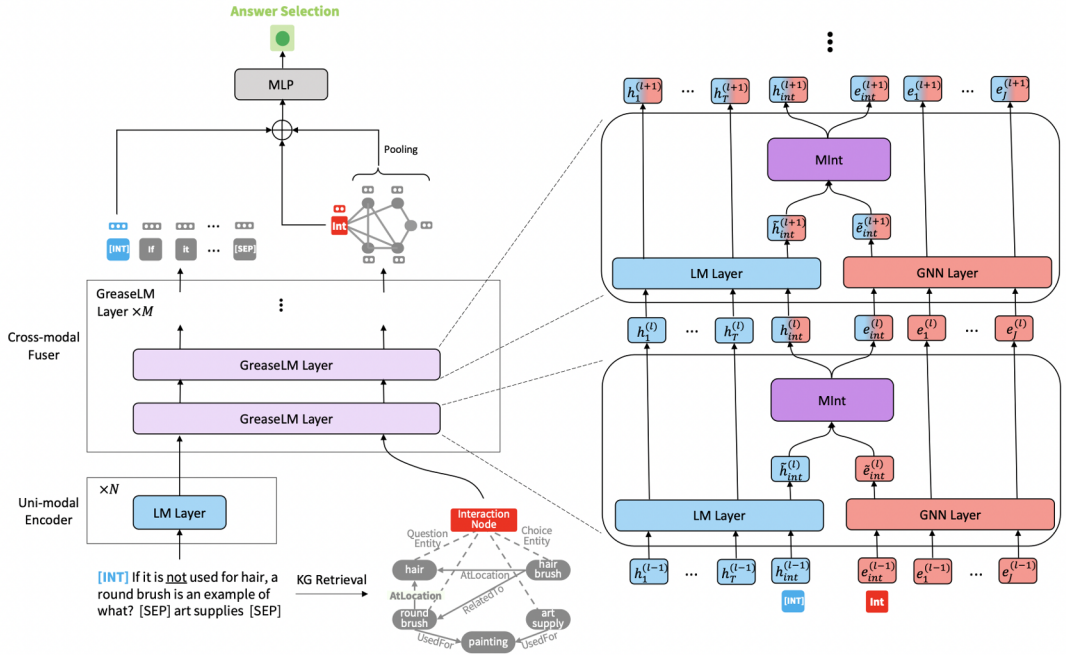


Figure 2: The Overall Architecture of Heterogeneous Graph Transformer. Given a sampled heterogeneous sub-graph with t as the target node, s_1 & s_2 as source nodes, the HGT model takes its edges $e_1 = (s_1, t)$ & $e_2 = (s_2, t)$ and their corresponding meta relations $< \tau(s_1), \phi(e_1), \tau(t) >$ & $< \tau(s_2), \phi(e_2), \tau(t) >$ as input to learn a contextualized representation $H^{(L)}$ for each node, which can be used for downstream tasks. Color decodes the node type. HGT includes three components: (1) meta relation-aware heterogeneous mutual attention, (2) heterogeneous message passing from source nodes, and (3) target-specific heterogeneous message aggregation.

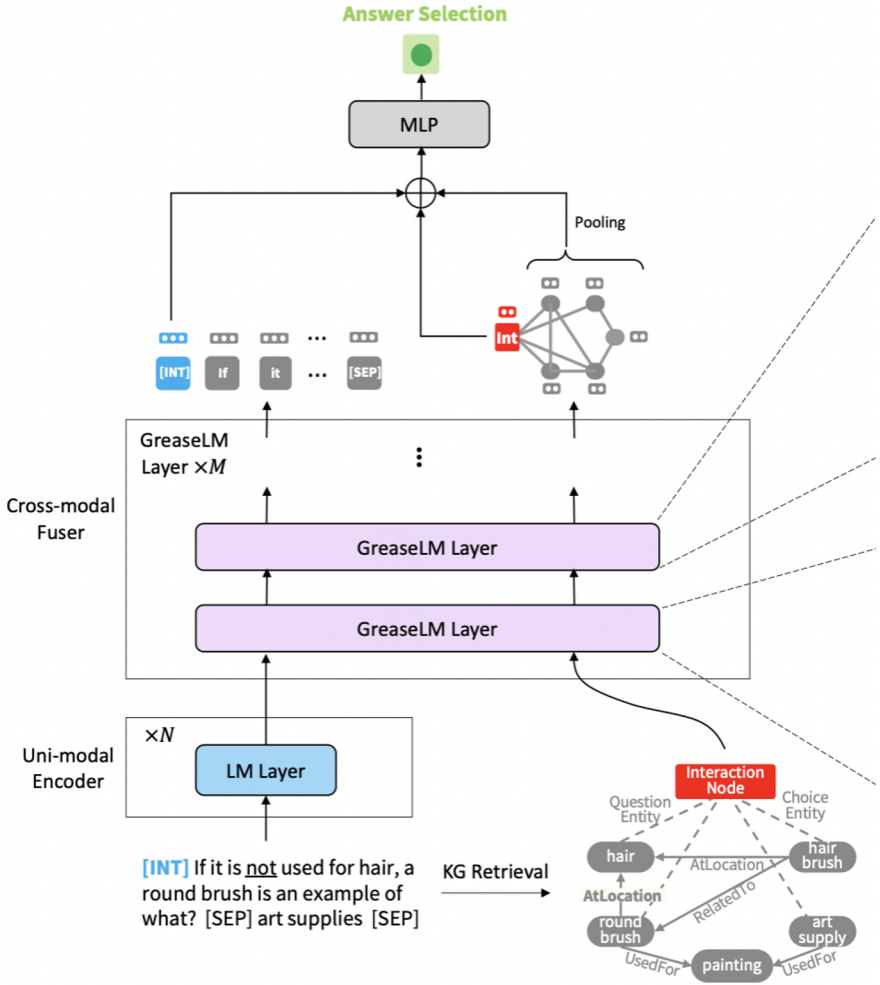
Hu, Ziniu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. "Heterogeneous graph transformer." In *Proceedings of The Web Conference 2020*, pp. 2704-2710. 2020. https://dl.acm.org/doi/pdf/10.1145/3366423.3380027

GreaseLM: Graph REASoning Enhanced Language Models for Question Answering



Zhang, Xikun, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D. Manning, and Jure Leskovec. "GreaseLM: Graph REASoning Enhanced Language Models for Question Answering." *arXiv preprint arXiv:2201.08860* (2022). https://arxiv.org/pdf/2201.08860

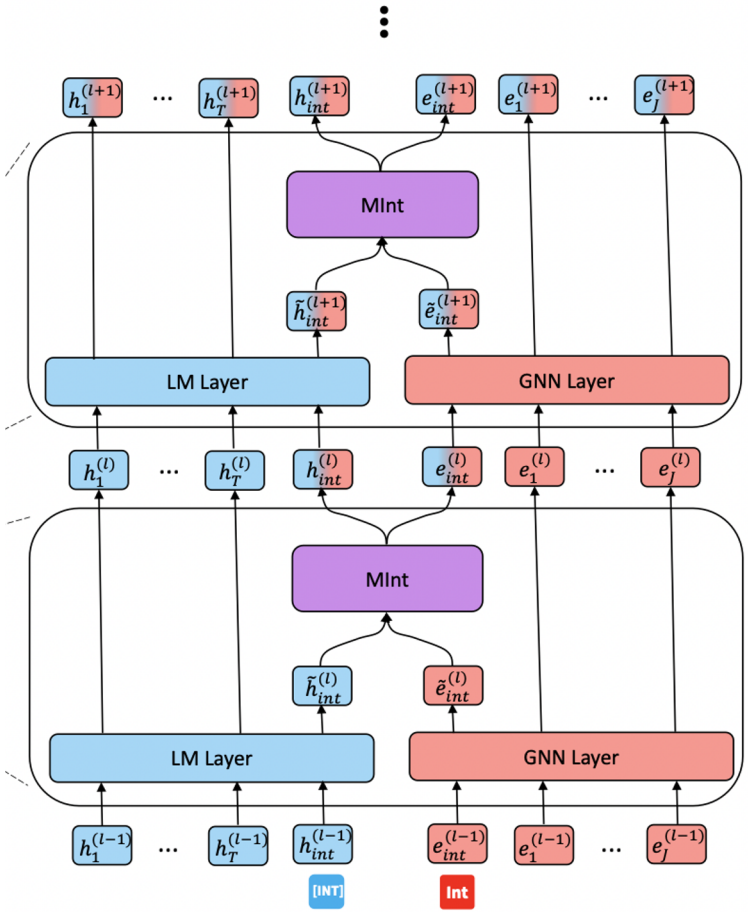
GreaseLM


Answer Selection
MLP
Pooling
[INT]
If
it
[SEP]
Int
GreaseLM Layer ×M
Cross-modal Fuser
GreaseLM Layer
GreaseLM Layer
×N
Uni-modal Encoder
LM Layer
[INT] If it is not used for hair, a round brush is an example of what? [SEP] art supplies [SEP]
KG Retrieval
Interaction Node
Question Entity
Choice Entity
hair
AtLocation
hair brush
AtLocation
RelatedTo
round brush
art supply
UsedFor
painting
UsedFor

GreaseLM

GNN corresponds to the **GAT** layer

LM model stands for **RoBERTa-Large / AristoRoBERTa / SapBERT**



Do Transformers Really Perform Bad for Graph Representation?

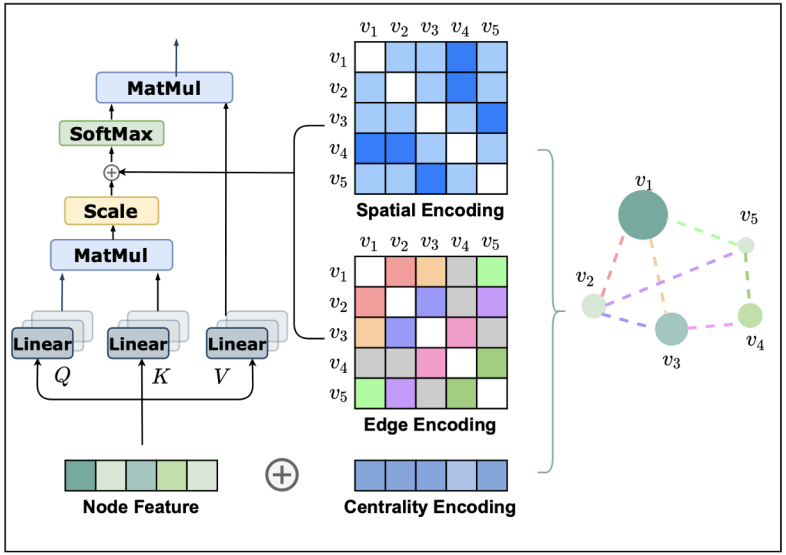


Figure 1: An illustration of our proposed centrality encoding, spatial encoding, and edge encoding in Graphormer.

Conclusion

- Transformers are also used for graph structures
- Many ways to present nodes and edges as input
- More graph transformers: https://github.com/ChandlerBang/awesome-graph-transformer