Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI
Humanities-Centered AI

Transformer-based Algorithmics

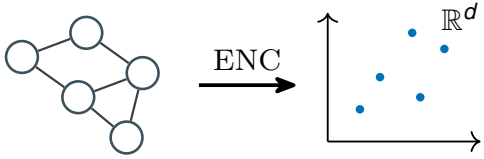
Malte Luttermann – Institute for Humanities-Centered Artificial Intelligence (CHAI)

June 24, 2026

Version from June 23, 2026

Topics for Today

- Node embeddings
- Graph neural networks (GNNs)
- Graph convolutional networks (GCNs) and GraphSAGE
- (Sub)graph embeddings



Node Embeddings – Idea

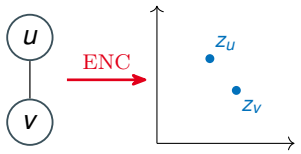
Goal

Map every node $v \in V$ to a vector $z_v \in \mathbb{R}^d$ such that **similarity in the graph** is reflected by **similarity in the embedding space**.

Encoder. $\mathrm{ENC}\colon V \to \mathbb{R}^d,\ v \mapsto z_v$.

Decoder / similarity. Compare embeddings, e.g. via the dot product:

$$\mathrm{sim}(u, v) \approx z_u^\top z_v.$$

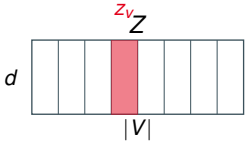


Shallow Embedding – Lookup Matrix

Shallow encoder. Embedding is just a lookup in a matrix:

$$\mathrm{ENC}(v) = Z \cdot \mathbf{1}_v, \qquad Z \in \mathbb{R}^{d \times |V|}.$$

- One **column per node** – learned directly.
- Examples: DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), node2vec (Grover and Leskovec 2016), LINE (Tang et al. 2015).
- Training objective rewards $z_u^\top z_v$ being large for nodes that are **close in the graph** (adjacent, share neighbours, co-occur on a random walk).



Exercise 1 – Limitations of Shallow Embeddings

The embedding matrix $Z \in \mathbb{R}^{d \times |V|}$ stores one independently-learned vector per node.

Discussion. Which problems does this design have? Think about...

- the **number of parameters** as $|V|$ grows,
- what happens for nodes **not seen during training**,
- the **node features** that real graphs carry (text, images, attributes, ...).

Exercise 1 – Solution

1. Parameter count. $\Theta(|V| \cdot d)$ parameters. No sharing between nodes – each z_v learned in isolation.

2. Transductive only. Z has no column for an unseen node $\Rightarrow$ cannot embed nodes that arrive after training, nor new graphs.

3. Ignores node features. Two nodes with very different attributes can land anywhere in $\mathbb{R}^d$; structural information alone drives the embedding.

Way out. Build the embedding from a node's **local neighbourhood** and its features $\Rightarrow$ *Graph Neural Networks*.

Setup

A graph $G = (V, E)$ with

- vertex set V and adjacency matrix $A \in \{0, 1\}^{|V| \times |V|}$,
- a feature matrix $X \in \mathbb{R}^{|V| \times f}$ – one row per node.

Key idea

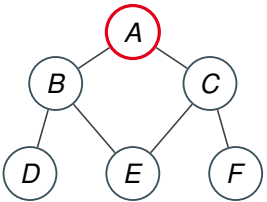
Embed a node by **aggregating information from its neighbours**, recursively. Every node v has its own **computation graph**, determined by A.

Features x_v can be: one-hot identifiers, degree / clustering statistics, text or image embeddings, profile attributes, ...

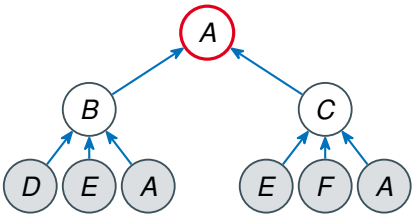
Neighbourhood Aggregation

Computation graph for a target node *v*. Unroll the neighbourhood layer by layer.

Input graph



Two-layer computation graph for *A*



Each node has the **same aggregation function** – weights are shared across positions.

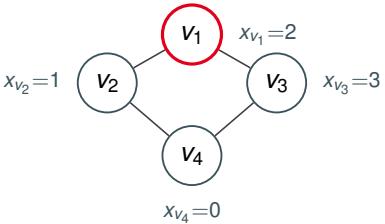
A Single GNN Layer

Update rule (basic form)

$$h_v^{(k)} = \sigma\left(W_k \cdot \frac{1}{|N(v)|} \sum_{u \in N(v)} h_u^{(k-1)} \; + \; B_k \cdot h_v^{(k-1)} \right)$$

- $h_v^{(0)} = x_v$ (node features), $h_v^{(K)} = z_v$ (final embedding).
- W_k: weight on the **aggregated neighbour message**.
- B_k: weight on the **node's own previous state**.
- σ: non-linearity (e.g. ReLU, where $\mathrm{ReLU}(x) = \max\{0, x\}$).
- Same W_k, B_k for **each node** at layer $k \Rightarrow$ parameters scale with K, not with $|V|$.

Exercise 2 – One Layer of Aggregation



Setting. Scalar features, $h_v^{(0)} = x_v$. One GNN layer:

$$h_v^{(1)} = \mathrm{ReLU}\big(W\,\bar{h}_v + B\,h_v^{(0)}\big),$$

$$\bar{h}_v = \tfrac{1}{|N(v)|}\sum_{u\in N(v)} h_u^{(0)}, \quad W{=}\tfrac{1}{2},\ B{=}1.$$

Tasks.

(1) Compute $h_{v_1}^{(1)}$.

(2) Compute $h_{v_4}^{(1)}$ and compare.

(3) When does v_4 reach v_1 – after one layer, or only after two?

Exercise 2 – Solution

(1) For v_1. $N(v_1) = \{v_2, v_3\}$, so

$$\bar{h} = \tfrac{1}{2}(1+3) = 2.$$

$$h_{v_1}^{(1)} = \mathrm{ReLU}\big(\tfrac{1}{2}\cdot 2 + 1\cdot 2\big) = \mathrm{ReLU}(3) = \mathbf{3}.$$

(2) For v_4. $N(v_4) = \{v_2, v_3\}$, $\bar{h} = 2$, $h_{v_4}^{(0)} = 0$.

$$h_{v_4}^{(1)} = \mathrm{ReLU}\big(\tfrac{1}{2}\cdot 2 + 1\cdot 0\big) = \mathbf{1}.$$

Same neighbours, different result. v_1 and v_4 share the same neighbour set, but their own features differ – the $B \cdot h_v^{(k-1)}$ term keeps the node »itself« visible.

(3) Receptive field of v_1.

- After 1 layer: $\{v_1, v_2, v_3\}$.
- After 2 layers: also v_4 – reached via v_2 or v_3.

K layers $\Rightarrow$ K-hop receptive field.

Graph Convolutional Networks (GCN)

Kipf and Welling (2017). One matrix for self and neighbours, normalised by degree:

$$h_v^{(k)} = \sigma\left(W_k \sum_{u \in N(v) \cup \{v\}} \frac{h_u^{(k-1)}}{\sqrt{|N(u)|\,|N(v)|}} \right)$$

Two design choices.

- **Same** W_k for self and neighbours $\Rightarrow$ more parameter sharing.
- **Symmetric** per-neighbour normalisation $\Rightarrow$ down-weights high-degree neighbours.

Why the normalisation? A node connected to a hub would otherwise be dominated by the hub's message. The factor $1/\sqrt{|N(u)|\,|N(v)|}$ dampens this without fully washing it out.

GraphSAGE – Sample and Aggregate

Hamilton, Ying, and Leskovec (2017). Separately transform the **aggregated neighbours** and the **self embedding**, then **concatenate**:

$$h_v^{(k)} = \sigma\Big(\big[\, A_k \cdot \mathrm{AGG}\big(\{h_u^{(k-1)}, \forall u \in N(v)\}\big)\ ,\ B_k \cdot h_v^{(k-1)}\ \big]\Big)$$

Choices for AGG.

- **Mean / max**: Element-wise reduction (cheap, permutation-invariant).
- **Pool**: Multi-layer perceptron (MLP) on each neighbour, then element-wise max.
- **LSTM**: Apply to a random permutation of $N(v)$ (expressive, but *not* permutation-invariant).
- **Attention**: Weighted sum learned per neighbour – Graph Attention Networks (Veličković et al. 2018).

Exercise 3 – Picking a Strategy for Bot Detection

Scenario. We want to embed users of a social network for **bot detection**. Some users have **millions of followers**, others only a handful. New users join every day.

Discuss.

(1) Would **shallow embeddings (node2vec)** work here? Why or why not?

(2) For a **GCN**: how does its normalisation help, and what does it cost us at high-degree nodes?

(3) For **GraphSAGE**: which aggregator would you pick – mean, pool, or LSTM – and what is the trade-off?

(4) Why do we usually sample a fixed-size neighbourhood per node?

Exercise 3 – Discussion Points

1. Shallow embeddings. Transductive – no embedding for users who join after training. $\Theta(|V| \cdot d)$ parameters do not scale.

2. GCN normalisation. $1/\sqrt{|N(u)|\,|N(v)|}$ stops a follower of a hub from being drowned out by the hub. Cost: messages from very popular accounts are heavily down-weighted – influential signal can be lost.

3. GraphSAGE aggregator.

- Mean: cheap, robust, ignores neighbour ordering.
- Pool: more expressive (per-neighbour MLP) but more parameters.
- LSTM: most expressive, but result depends on the order $\Rightarrow$ train on random permutations.

4. Sampling $N(v)$. Hubs would otherwise blow up the message volume. Fixed-size sampling keeps cost per node **predictable** and provides a regularizer.

From Nodes to (Sub)graphs

Question. Given node embeddings z_v, how do we obtain an embedding of a whole **subgraph** $S \subseteq V$ (or the entire graph)?

From Nodes to (Sub)graphs

Question. Given node embeddings z_v, how do we obtain an embedding of a whole **subgraph** $S \subseteq V$ (or the entire graph)?

(a) Sum or average.

$$z_S = \sum_{v \in S} z_v \quad \text{or} \quad z_S = \frac{1}{|S|} \sum_{v \in S} z_v$$

Simple, permutation-invariant – but the sum can grow without bound and may blur fine structure.

(b) Virtual node. Add a node v_S connected to every $v \in S$, run the GNN, read off z_{v_S}.

(c) Hierarchical pooling. Cluster the graph, embed each cluster, repeat – the analog of »pooling« in convolutional neural networks (CNNs).

Exercise 4 – Choosing a Subgraph Strategy

For each task below, discuss which of **sum / mean**, **virtual node**, or **hierarchical pooling** you would pick – and why.

(1) Predict the **toxicity of a molecule**, given as a small graph (10-50 atoms) with bond annotations.

(2) Classify the **role of a community** (e.g., »spammers«, »researchers«) inside a large social network of millions of users.

(3) Embed an entire **knowledge graph** so we can compare two whole KGs.

Exercise 4 – Discussion Points

(1) **Molecule toxicity.** Small graph, fine structure matters (e.g., a single functional group decides toxicity). **Hierarchical pooling** preserves substructures; a virtual node tends to over-smooth.

(2) **Community role.** $|S|$ varies wildly between communities. **Mean** (not sum) keeps the embedding scale comparable; a virtual node also works if it is added before training.

(3) **Whole knowledge graph.** Sum / mean over millions of nodes loses too much. **Hierarchical pooling** – or a learned readout – preserves structure at multiple scales.

References I

- Grover, Aditya and Jure Leskovec (2016). node2vec: Scalable Feature Learning for Networks. In: *Proceedings of the Twenty-Second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2016)*. Association for Computing Machinery, 855–864.
- Hamilton, William L., Rex Ying, and Jure Leskovec (2017). Inductive Representation Learning on Large Graphs. In: *Advances in Neural Information Processing Systems 30 (NeurIPS-2017)*. Curran Associates, Inc. 1024–1034.
- Kipf, Thomas N. and Max Welling (2017). Semi-Supervised Classification with Graph Convolutional Networks. In: *Fifth International Conference on Learning Representations (ICLR-2017)*. OpenReview.net.
- Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena (2014). DeepWalk: Online Learning of Social Representations. In: *Proceedings of the Twentieth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2014)*. Association for Computing Machinery, 701–710.

References II


Tang, Jian, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei (2015). LINE: Large-scale Information Network Embedding. In: *Proceedings of the Twenty-Fourth International Conference on World Wide Web (WWW-2015)*. International World Wide Web Conferences Steering Committee, 1067–1077.


Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio (2018). Graph Attention Networks. In: *Sixth International Conference on Learning Representations (ICLR-2018)*. OpenReview.net.