Attention and Transformers

StructML – Tutorial 9

- Graph Attention Networks (GAT)
- Q-K-V Attention mechanism
- Transformer building blocks
- Transformers for Heterogenous Graphs
  - Tuple level - Heterogenous Graph Transformers (2020)
  - Cell level - Relational Transformers (2025)

Graph Attention Networks (GAT)

- Idea: some neighbors are more relevant than others
  - Learn their importance

K attention heads

$$e_{ij} = a(\boldsymbol{W}h_i, \boldsymbol{W}h_j)$$

a – attention score between 2 vectors (Linear NN +ReLU)

$\boldsymbol{W}$- transform node features (shared)

Normalize attention scores among neighbors:

$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})}$$

Aggregation: weighted sum of neighbors

$$h_i' = \sigma\left(\sum_{j \in N_i} \alpha_{ij} \boldsymbol{W} h_j\right)$$

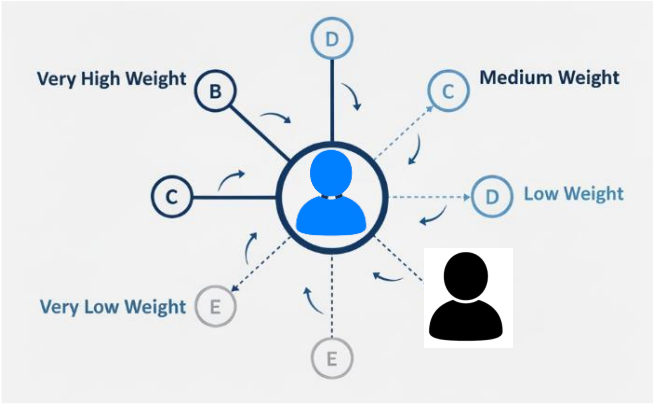
Neighborhood N_i includes i

$M^{(k)}$ - message function

$\alpha^{(k)}$ - aggregation

$U^{(k)}$ - update - identity

1 attention "head"



Aggregate the heads:

- Concatenate (in inner layers)
- Mean (for prediction head)

Veličković, Cucurull, et al. Graph attention networks. (2017).

GAT Properties

- Benefits:
  - **Dynamic weighting** – capture neighbor importance
  - **Inductive** – can easily handle new nodes/unseen graphs – attention scores do not depend on the global graph
  - (Somewhat) **interpretable** – the attention indicates what neighbors had a large impact on prediction
- Limitations:
  - **Homogeneous** – can't handle different node dimensions or edge semantics
  - **High computational cost** of attention for all edges (instead of simple aggregations)

- Graph Attention Networks (GAT)
- Q-K-V Attention mechanism
- Transformer building blocks
- Transformers for Heterogenous Graphs
  - Tuple level - Heterogenous Graph Transformers (2020)
  - Cell level - Relational Transformers (2025)

Attention as Queries, Keys, and Values

- Attention is basically a weighted average:

$$\sum \alpha_{ij} h_j$$

α_{ij} - learned weight of neighbor j for node i

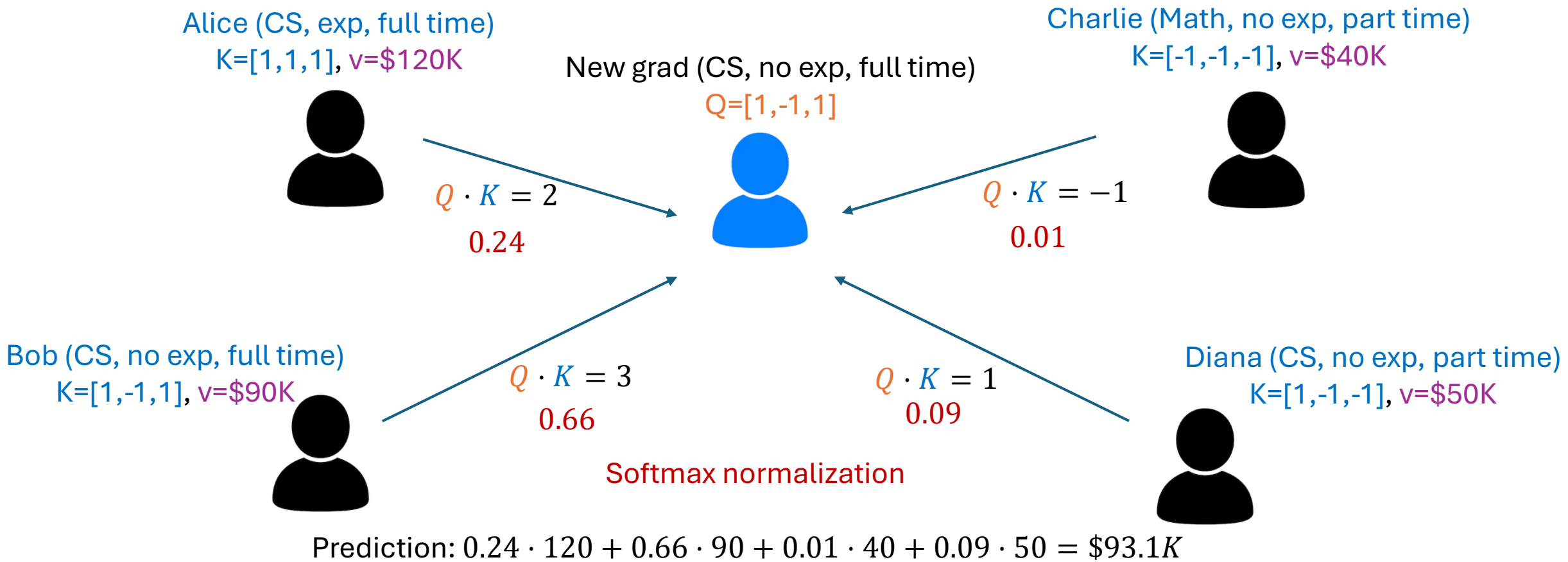
h_j - info (features/representation) of neighbor j.

- But weights may depend on different aspects of the connection between the node and its neighbors.

Attention as Queries, Keys, and Values: Example

Example: predict salary for a BSc graduate.

Features: degree (CS/Math), experience (exp/no exp), job type (full time/part time)



Prediction: $0.24 \cdot 120 + 0.66 \cdot 90 + 0.01 \cdot 40 + 0.09 \cdot 50 = \$93.1K$

Attention as Queries, Keys and Values

- In the general case: Q, K, V are **learned** representations
  - $Q \in \mathbb{R}^{1\times d_k}$ – representing the center (target) node u
  - $\mathrm{K} \in \mathbb{R}^{m\times d_k}$ – representing features of the neighboring nodes $v \in N(u)$
  - $V \in \mathbb{R}^{m\times d_v}$ – representing the information each neighbor holds
- Computed via shared transformations:

$$Q = h_u \cdot W_Q \qquad K = h_v \cdot W_K \qquad V = h_v \cdot W_V$$

- Attention is given by:

$$\frac{Q \cdot K^\top}{\sqrt{d_k}} \cdot V$$

Scale by key dimension.
Without it: high variance
→Softmax "peaks"
→vanishing gradients

- Usually – multiple heads (different $W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}$ per head).
  - Resulting representations (from each head) are concatenated.

Compared to GAT:

- One linear projection (no Q,K,V view)
- Attention score was:

$\mathrm{MLP}(\boldsymbol{W}h_u \oplus \boldsymbol{W}h_v)$

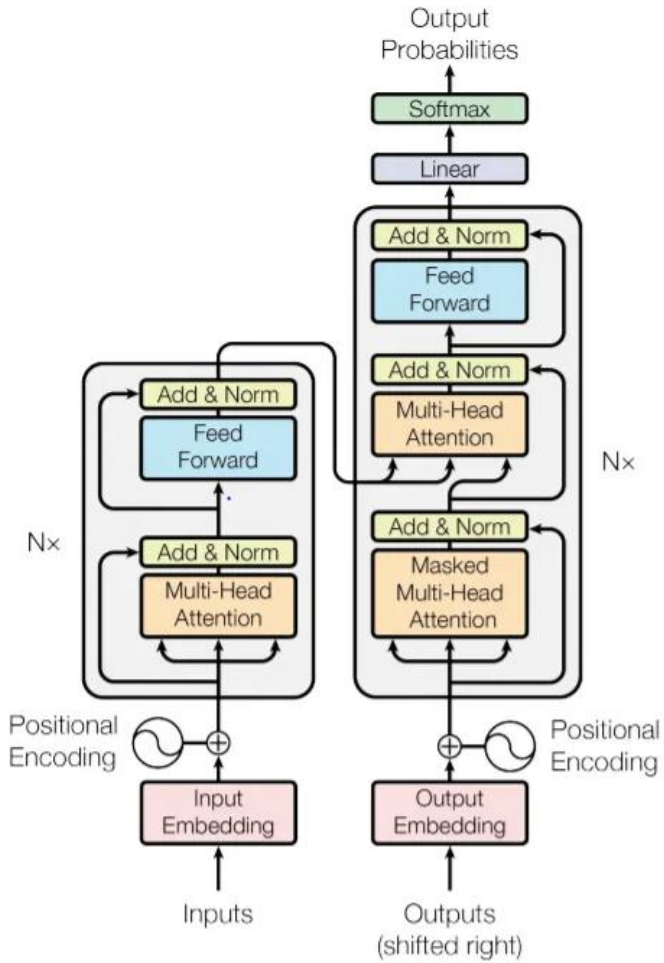
- *Update* did not contain learned parameters – but it can (we will see)

Transformers for NLP – Basic Building Block

- Used for generating a **sequence** of text
  - E.g., translation task, summarization, etc.
- Each input token is embedded, and each position index is embedded (to capture order).
- Encoder: several transformer layers, using:

  self attention between input tokens (~words)

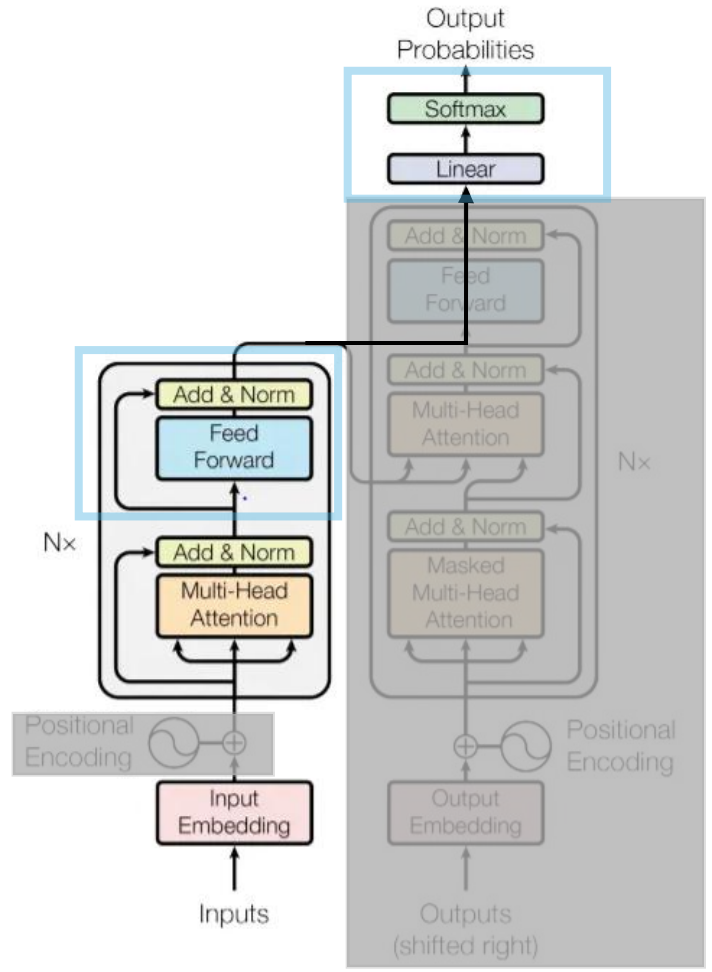
  + feed forward NN
- Decoder: more transformer layers, using:
  - self attention between output tokens
  - + cross attention to input tokens
  - + feed forward NN
- Prediction head



Self attention – observer and context are from the same "world"
Cross attention – observer and context are from separate "worlds"

Transformers for Relational Databases

- No elaborate decoder
  - Prediction tasks usually require a single output value, not a sequence
- No positional encoding
  - The tokens are rows or cells, not an ordered sequence (sentence)
- What we do with the attention output may change (in different architectures).



- Graph Attention Networks (GAT)
- Q-K-V Attention mechanism
- Transformer building blocks
- Transformers for Heterogenous Graphs
  - Tuple level - Heterogenous Graph Transformers (2020)
  - Cell level - Relational Transformers (2025)

Homogeneous ~~Heterogeneous~~ Graph Transformers (HGT)

$$Q_t = H_t W_Q \qquad K_s = H_s W_K \qquad V_s = H_s W_V$$

$1 \times 5 \quad 5 \times d_k$ $\qquad$ $1 \times 12 \neq 5 \times d_k$ Because W_Q, W_K must have the same dimensions!

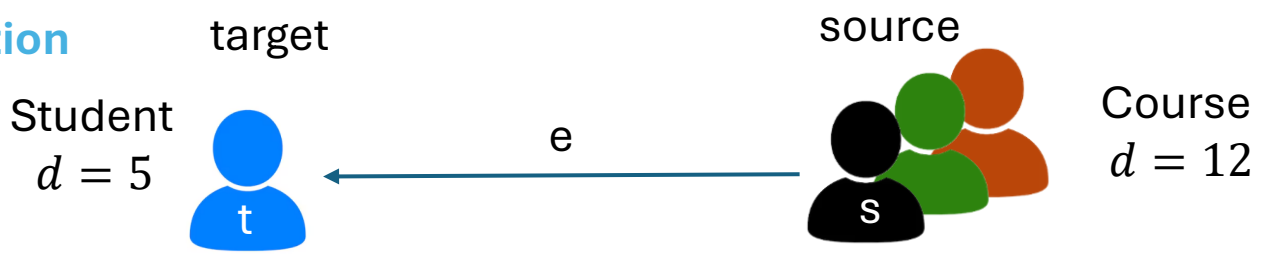
$$\mathrm{Att}_{t,s} = \frac{Q_t K_s^\top}{\sqrt{d}}$$

Aggregation: $$H_t^{new} = \sum_{s \in N(t)} \mathrm{softmax}\left(\mathrm{Att}_{t,s}\right) \cdot V_s$$

But what if H_t, H_s have different dimensions?
Can't perform all of these: $H_t W_Q$, $H_s W_K$, $Q_t K_s^\top$

Heterogeneous Graph Transformers (HGT)

Heterogeneous mutual attention



$$Q_t = H_t W_Q^{\tau(t)} \qquad K_s = H_s W_K^{\tau(s)} \qquad V_s = H_s W_V^{\tau(s)}$$

Projection to shared space:
Different $W_Q^{\tau(t)}, W_K^{\tau(s)}, W_V^{\tau(s)}$ for each node type $\tau(\cdot)$.

$$\text{Att}_{t,e,s} = \frac{Q_t W_{ATT}^{\phi(e)} K_s^\top}{\sqrt{d_k}} \cdot \mu_{\tau(t),\phi(e),\tau(s)}$$

Edge-type scale:
Different edges have different prior importance scores.

Edge-type specific weight matrix:
Different edges have different meaning.

Student $\xrightarrow{\text{takes}}$ course

vs

Student $\xrightarrow{\text{teaches}}$ course

Aggregation:

$$H_t' = \sum_{s \in N(t)} \text{softmax}\left(\text{Att}_{t,s}\right) \cdot V_s \cdot W_{MSG}^{\phi(e)}$$

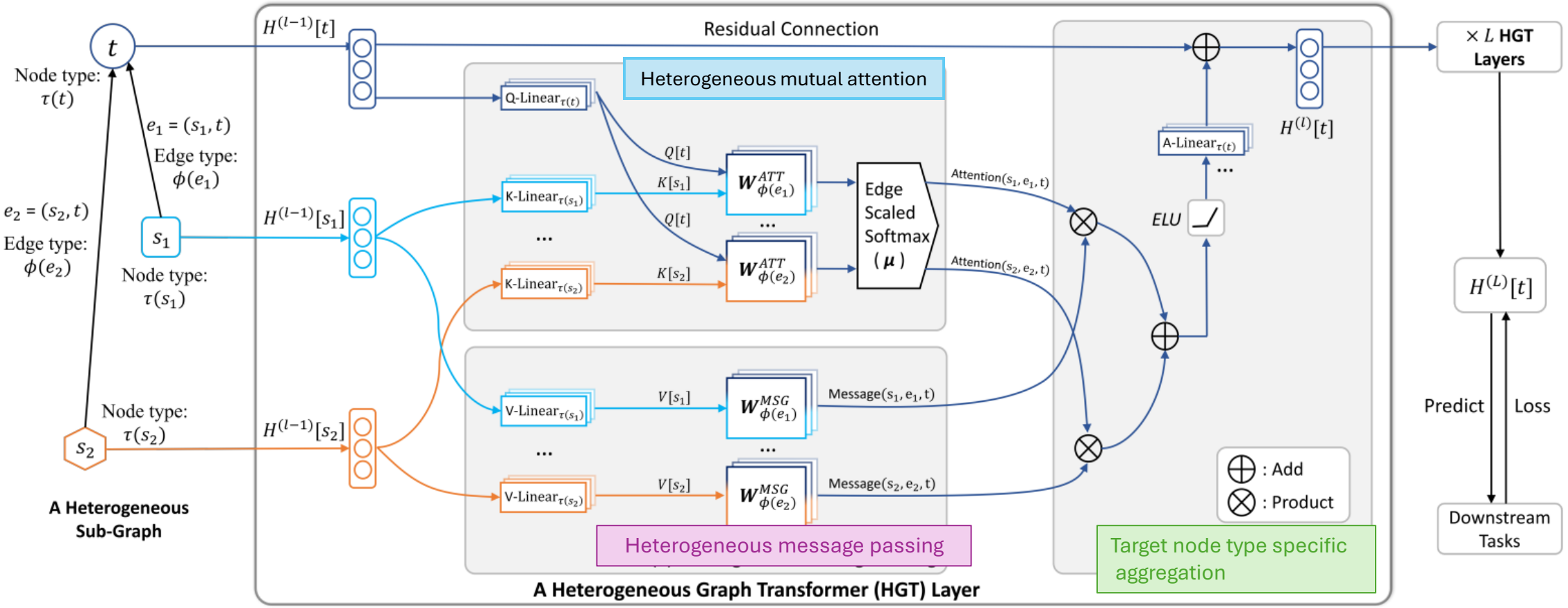
Heterogeneous message passing:
The message (value) is multiplied by an edge type specific matrix

$$H_t^{new} = W_A^{\tau(t)} \sigma(H_t') + H_t$$

Node type specific aggregation

What's this?

Heterogeneous Graph Transformers (HGT)



Hu et al. Heterogeneous graph transformer. *WWW 2020.*

HGT paper (partial) results

HGT has many more parameters than GAT, GCN
Most are due to heterogeneity.
Similar to RGCN size.

Paper settings:
3 layers
8 attention heads
256 hidden dim.

GNN Models		GCN [9]	RGCN [14]	GAT [22]	HetGNN [27]	HAN [23]	HGT$^{+RTE}_{-Heter}$	HGT$^{+RTE}_{+Heter}$
# of Parameters		1.69M	8.80M	1.69M	8.41M	9.45M	3.88M	8.20M
Batch Time		0.46s	1.24s	0.97s	1.35s	2.27s	1.14s	1.50s
Paper–Field (L_1)	NDCG	.608±.062	.603±.065	.622±.071	.612±.063	.618±.058	.689±.042	**.718±.014**
	MRR	.679±.069	.683±.056	.694±.065	.689±.060	.691±.051	.779±.027	**.823±.019**
Paper–Field (L_2)	NDCG	.344±.021	.322±.053	.357±.058	.346±.071	.352±.051	.371±.043	**.403±.041**
	MRR	.353±.053	.340±.061	.382±.057	.373±.051	.388±.065	.397±.064	**.439±.078**
Paper–Venue	NDCG	.406±.081	.412±.076	.437±.082	.431±.074	.449±.072	.461±.066	**.473±.054**
	MRR	.215±.066	.216±.105	.239±.089	.245±.069	.254±.074	.265±.090	**.288±.088**
Author Disambiguation	NDCG	.826±.039	.835±.042	.864±.051	.850±.056	.859±.053	.875±.046	**.894±.034**
	MRR	.661±.045	.665±.054	.694±.052	.668±.061	.688±.049	.712±.032	**.732±.038**

HGT performance improvement (not only due to # parameters - see RGCN)

GCN and RGCN have similar performance, despite added parameters

GAT improves a little more

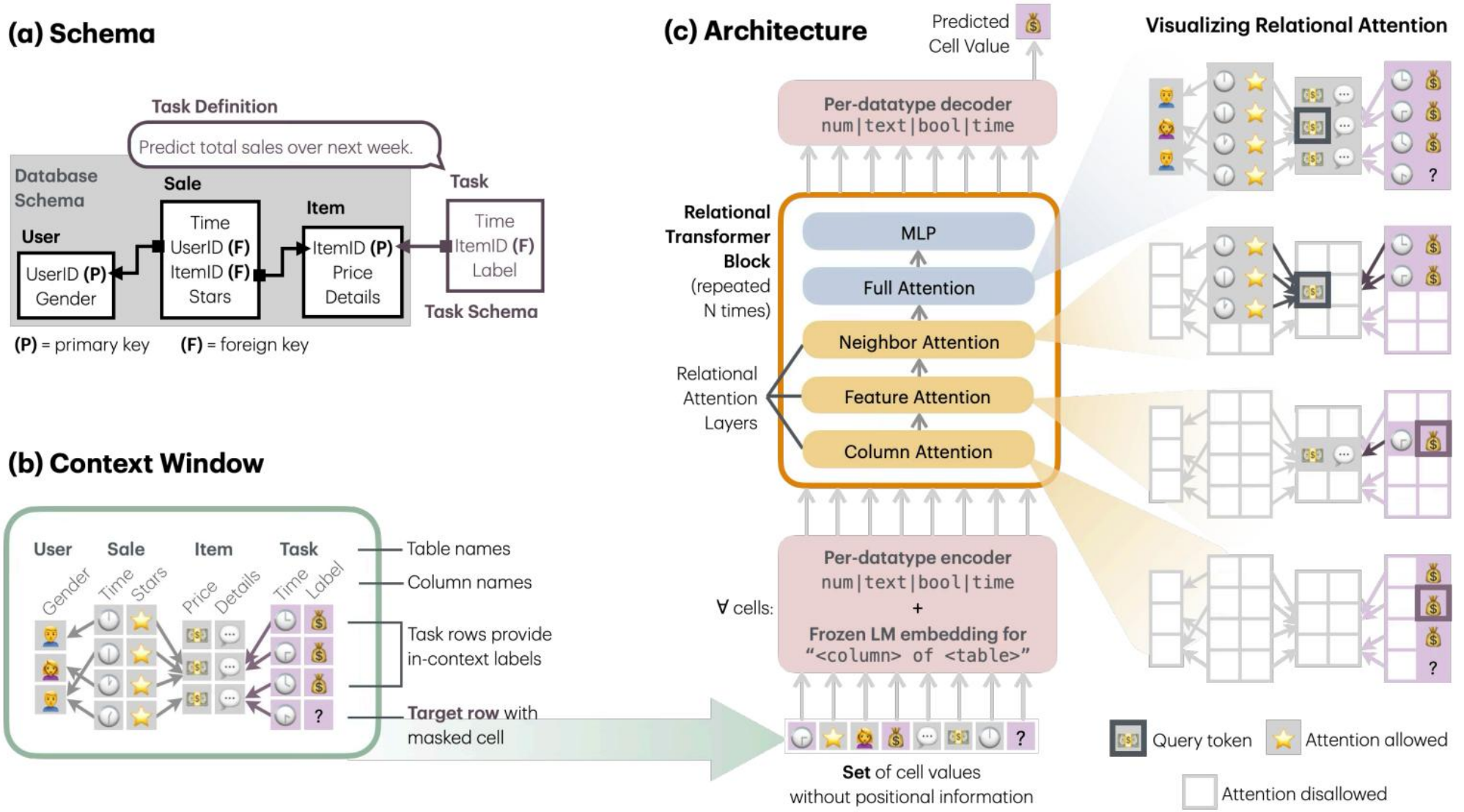
- Graph Attention Networks (GAT)
- Q-K-V Attention mechanism
- Transformer building blocks
- Transformers for Heterogenous Graphs
  - Tuple level - Heterogenous Graph Transformers (2020)
  - Cell level - Relational Transformers (2025)

Relational Transformers

- Operates on a cell-level instead of row-level.
  - Tokens are cells, not tuples.
- Trained on masked token prediction tasks
  - Can express autocomplete (e.g., missing values) or forecasting tasks.
- We will discuss:
  - How cells are encoded
  - How a neighborhood (context window) is fetched
  - How attention is used

Ranjan et al. Relational Transformer: Toward Zero-Shot Foundation Models for Relational Data. *2025*

Relational Transformers



Relational Transformers: Cell Encoding

- A cell: (v, c, t)
  value column table
- Value v is transformed by type into value encoding r:
  - Numeric/Boolean/timestamp – normalized
  - Text – embedded to a vector using a text encoder
- Final encoding contains value and schema information

$$x = W_d \cdot r + W \cdot \mathrm{emb}(\text{"<c> of <t>"})$$

Specific to data type | Value encoding | shared | Embedded schema information

- Encoding masked* cells: (for prediction)

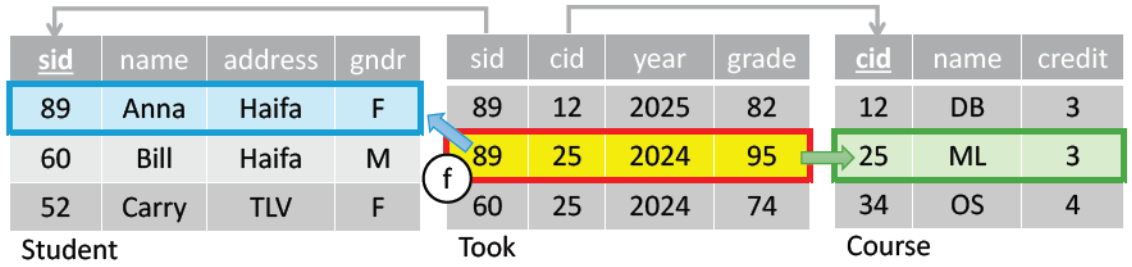
$$x = m_d + W \cdot \mathrm{emb}(\text{"<c> of <t>"})$$

Learned mask vector Per data type

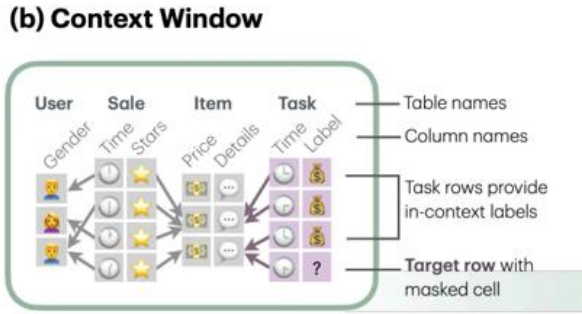
* First meaning of mask – hide a cell

Relational Transformers: Context Window

- From a given row f: BFS-like search, to get context of n cells.
- What tuples are added to the queue?
  - All parents of f (f has the FK)



  - A sample of children of f (f has the PK)
- Once a row is selected from the queue, all cells are added to the context.
- The context is a **set** of cells
  - Order is not important – no positional encoding required.



Relational Transformers: Relational Attention

- All cells are encoded to the same space –
  - Can share W^Q, W^K, W^V matrices
- Employ a masked* attention function:

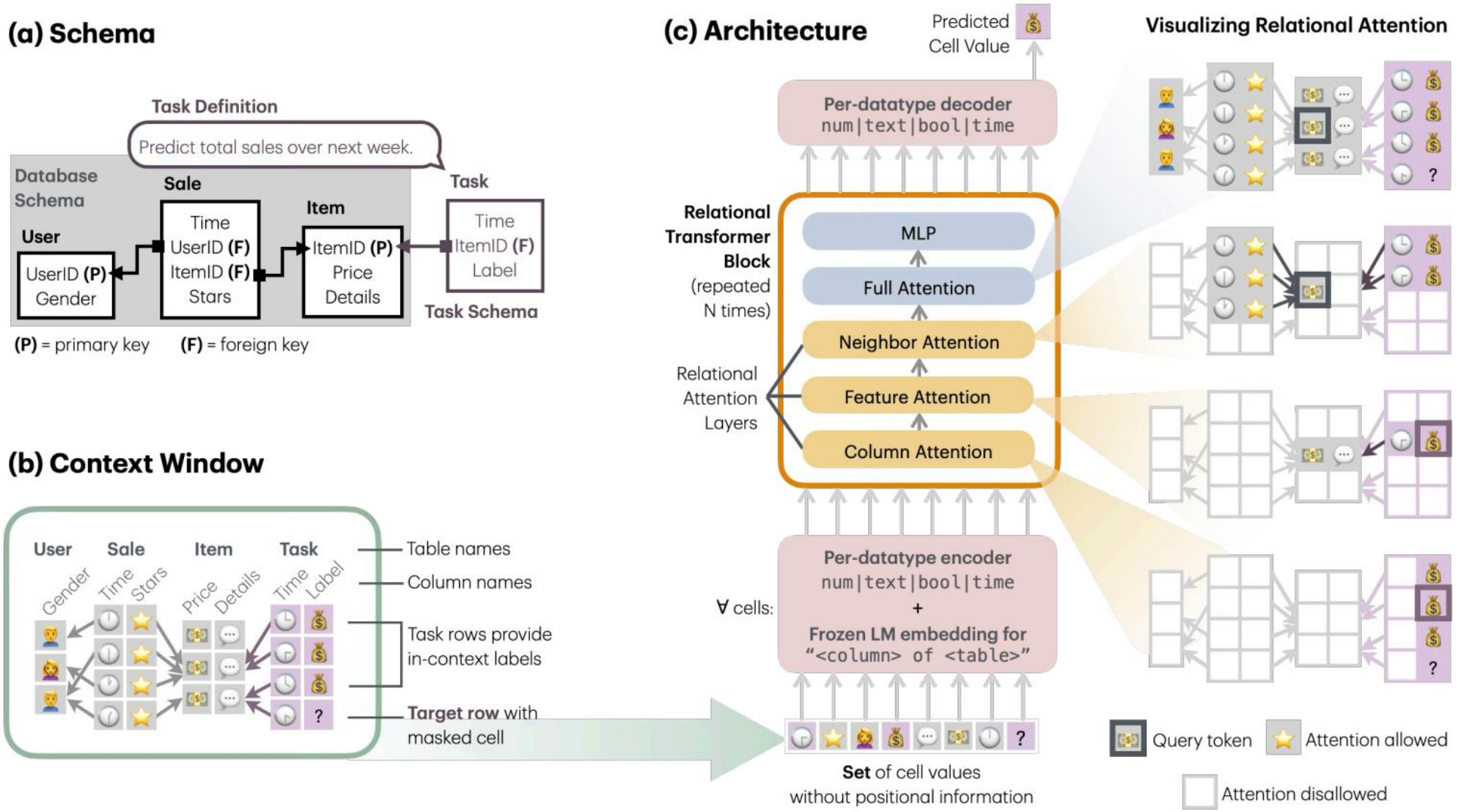
*Second meaning of mask – what cells are allowed to attend to each other

$$\mathrm{Att}(Q,K,V;M) = \mathrm{softmax}\left(\frac{\mathrm{Mask}(QK^T;M)}{\sqrt{d}}\right)\cdot V$$

$$\mathrm{Mask}(A;M)_{ij} = \begin{cases} A_{ij} & \text{if } M_{ij}=1 \\ -\infty & \text{if } M_{ij}=0 \end{cases}$$

  - If cells are "allowed" to see each other ($M_{ij}=1$), compute the attention as usual
- Different masks – different attention types:
  - Column attention – model value distribution
  - Feature attention – cells on same row or linked parent rows
  - Neighbor attention – cells on linked child rows
  - Full attention - all cells

Relational Transformers



Summary: Attention and Transformer Relational Models

Property	**Graph Attention (GAT)**	**Heterogeneous Transformers (HGT)**	**Relational Transformers (RT)**
Token level	Tuple	Tuple	Cell
Attention formula	$\alpha_{ij} = \text{softmax}\left(a\left(\boldsymbol{W}h_i, \boldsymbol{W}h_j\right)\right)$ $h_i' = \sigma\left(\sum_{j \in N_i} \alpha_{ij} \boldsymbol{W} h_j\right)$	$\text{softmax}\left(\frac{Q_t W_{ATT}^{\phi(e)} K_s^\top}{\sqrt{d}} \cdot \mu_{\tau(t),\phi(e),\tau(s)}\right) \cdot V_s \cdot W_{MSG}^{\phi(e)}$	$\text{softmax}\left(\frac{\text{Mask}(QK^T; M)}{\sqrt{d}}\right) \cdot V$
Heterogeneity	Not supported	Type specific node projection, edge attention, edge scaling, message passing, message aggregation	Type specific **tokenization**
Paper setting and Typical Size	1.69M	8.2M	22M(!)
Trained on	Downstream tasks	Downstream tasks	Masked token prediction