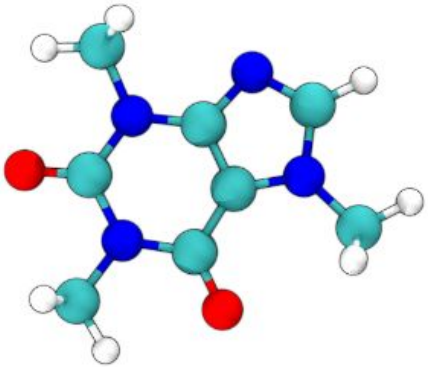
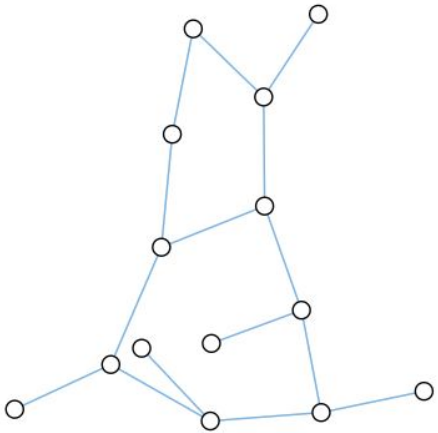
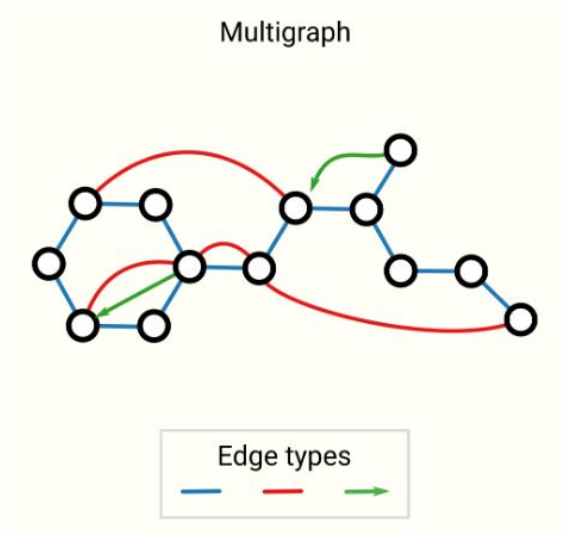
Do Transformers Really Perform Bad for Graph Representation?

Графовые данные

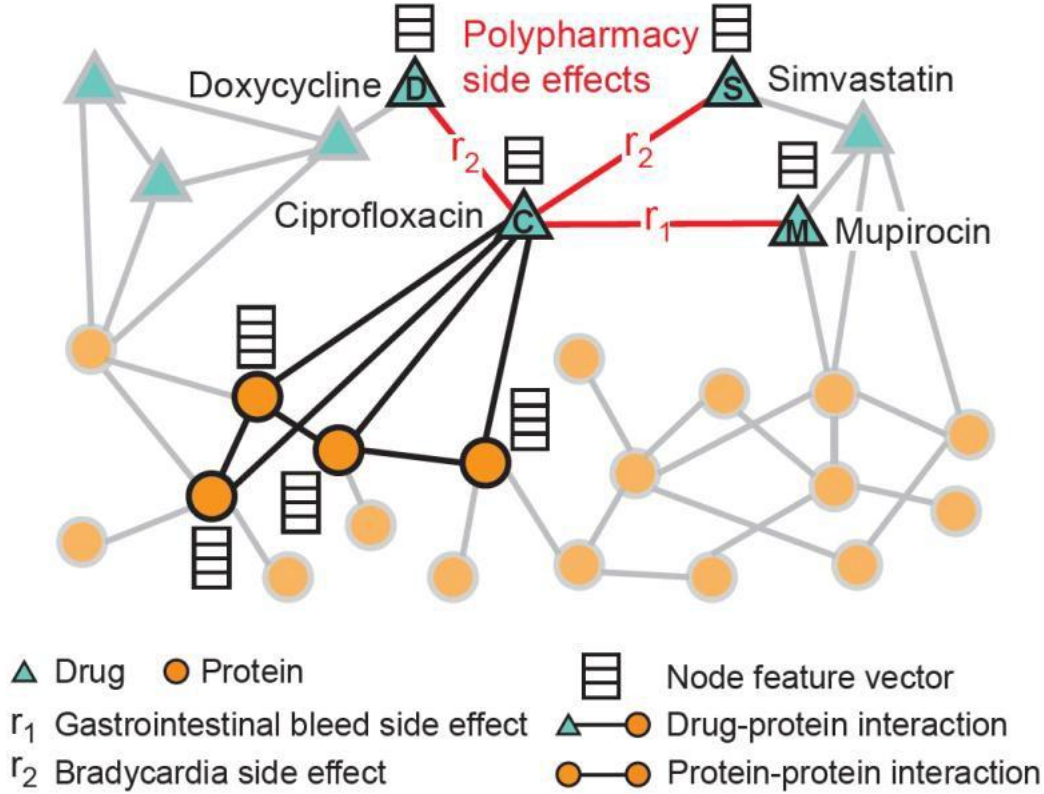
- Каждая вершина имеет признаковое описание в виде вектора
- В зависимости от данных ребра тоже могут иметь признаки





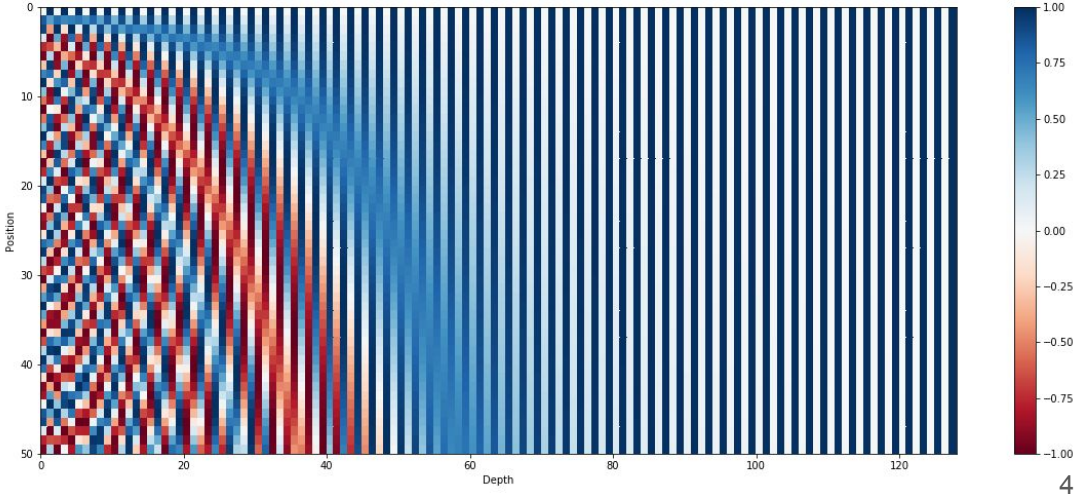


Графовые данные



В чем проблема трансформеров

- Трансформер работает с данными с простой структурой
- Трансформер использует positional encoding, непонятно как считать его для графов

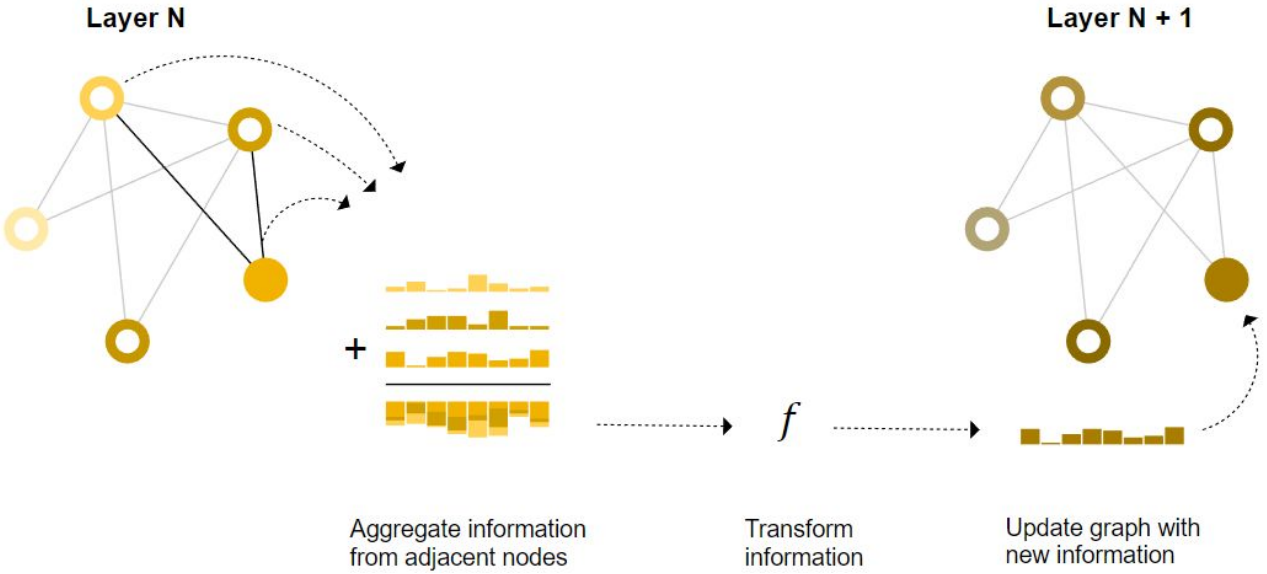


Как работают GNN

- GNN строят эмбеддинги для вершин/ребер учитывающие структуру графа которые затем подаются на вход нейросети
- Эмбеддинги строятся итеративно
- На каждой итерации два шага AGGREGATE и COMBINE.

Как работают GNN

- AGGREGATE собирает признаки из соседних вершин (берет среднее, сумму, максимум, …)
- COMBINE обновляет текущий вектор используя получившийся вектор



Graphormer

- Идея статьи - извлечь из графа информацию о позиционной структуре графа для использования в трансформере
- Для этого используют Centrality Encoding, Spatial Encoding и Edge Encoding

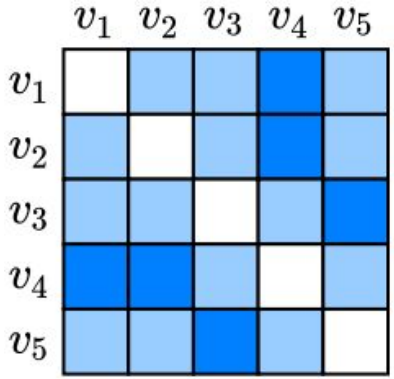
Centrality Encoding

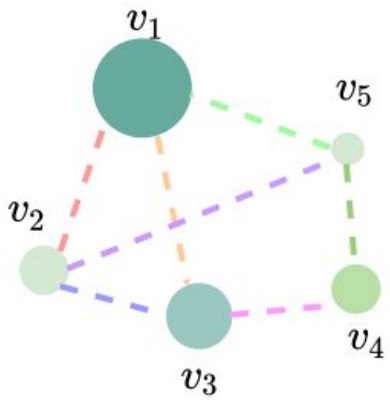
- Идея: вершины с высокой степенью важны
- Используем пару эмбеддингов для входной и выходной степени, прибавляем их признакам вершин
- Каждой степени (или диапазону степеней в зависимости от задачи) соответствует свой эмбеддинг
- В них текут градиенты, обучаются вместе с остальными весами

$$h_i^{(0)} = x_i + z^-_{\deg^-(v_i)} + z^+_{\deg^+(v_i)}$$

Spatial Encoding

- Для каждой пары вершин хотим хранить информацию о их взаимном расположении
- Используем длину кратчайшего пути
- Каждой длине пути соответствует обучаемый скаляр, он прибавляется к произведению query и key векторов в аттеншене

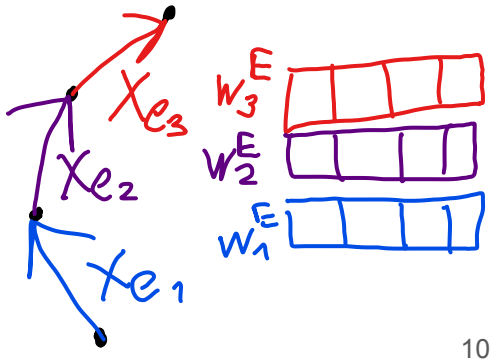




Edge Encoding in the Attention

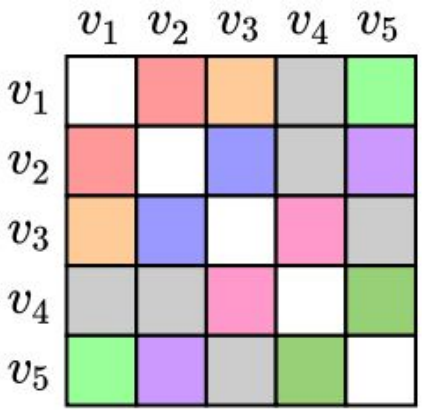
- Хотим использовать признаки ребер
- Для каждой пары вершин находим кратчайший путь между ними (один из)
- Идем вдоль этого пути и считаем скалярное произведение эмбеддингов ребер со специальными обучаемыми эмбеддингами w_n^E
- Каждому шагу (по порядку) соответствует свой эмбеддинг
- Усредняем получившиеся скаляры и прибавляем к произведению query и key векторов в аттеншене
- Эмбеддинги обучаемы

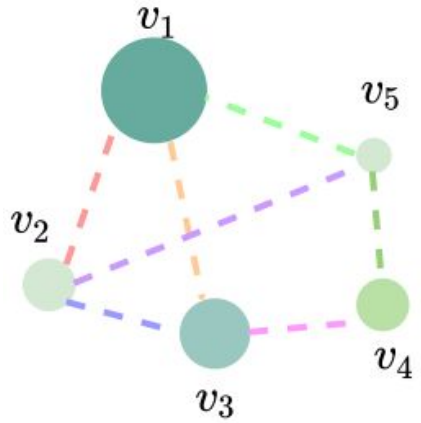
$$c_{ij} = \frac{1}{N} \sum_{n=1}^{N} x_{e_n} (w_n^E)^T$$



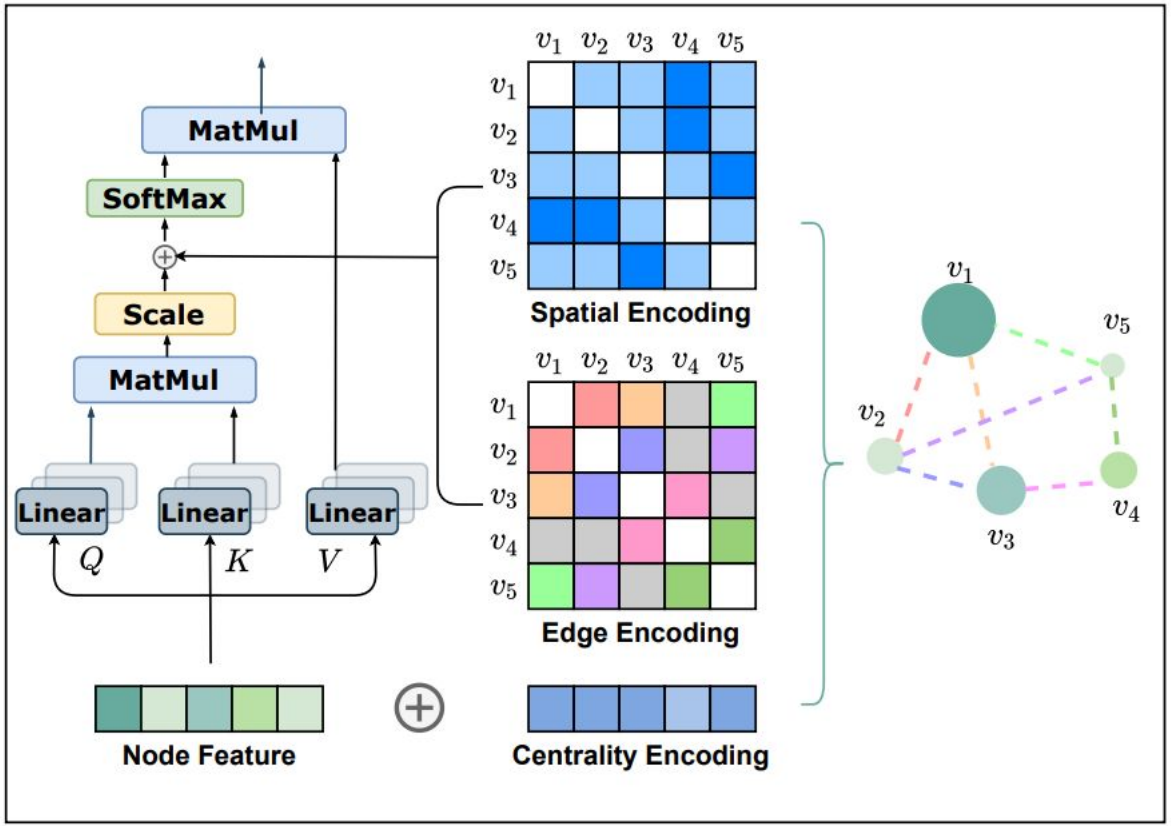
Edge Encoding in the Attention

- Идем вдоль этого пути и умножаем эмбеддинги ребер на специальные обучаемые эмбеддинги
- Каждому шагу (по порядку) соответствует свой эмбеддинг
- Усредняем получившиеся скаляры и прибавляем к произведению query и key векторов в аттеншене





Graphormer


MatMul
SoftMax
Scale
MatMul
Linear
Linear
Linear
Q
K
V
Node Feature
Spatial Encoding
Edge Encoding
Centrality Encoding

Graphormer

- Можно доказать что можно подобрать веса и функцию расстояния графформер будет повторять работу AGGREGATE и COMBINE шагов различных DNN

Graphormer

- Побили все соты на нескольких датасетах

method	#param.	train MAE	validate MAE
GCN [26]	2.0M	0.1318	0.1691 (0.1684*)
GIN [54]	3.8M	0.1203	0.1537 (0.1536*)
GCN-VN [26, 15]	4.9M	0.1225	0.1485 (0.1510*)
GIN-VN [54, 15]	6.7M	0.1150	0.1395 (0.1396*)
GINE-VN [5, 15]	13.2M	0.1248	0.1430
DeeperGCN-VN [30, 15]	25.5M	0.1059	0.1398
GT [13]	0.6M	0.0944	0.1400
GT-Wide [13]	83.2M	0.0955	0.1408
$\text{Graphormer}_{\text{SMALL}}$	12.5M	0.0778	0.1264
Graphormer	47.1M	**0.0582**	**0.1234**

Table 2: Results on MolPCBA.

method	#param.	AP (%)
DeeperGCN-VN+FLAG [30]	5.6M	28.42±0.43
DGN [2]	6.7M	28.85±0.30
GINE-VN [5]	6.1M	29.17±0.15
PHC-GNN [29]	1.7M	29.47±0.26
GINE-APPNP [5]	6.1M	29.79±0.30
GIN-VN[54] (fine-tune)	3.4M	29.02±0.17
Graphormer-FLAG	119.5M	**31.39**±0.32

Ablation study

Table 5: Ablation study results on PCQM4M-LSC dataset with different designs.

Node Relation Encoding		Centrality	Edge Encoding			valid MAE
Laplacian PE[13]	Spatial		via node	via Aggr	via attn bias(Eq.7)	
-	-	-	-	-	-	0.2276
✓	-	-	-	-	-	0.1483
-	✓	-	-	-	-	0.1427
-	✓	✓	-	-	-	0.1396
-	✓	✓	✓	-	-	0.1328
-	✓	✓	-	✓	-	0.1327
-	✓	✓	-	-	✓	0.1304