Графы знаний

Лекция 8 - Графовые нейросети для KGs

М. Галкин

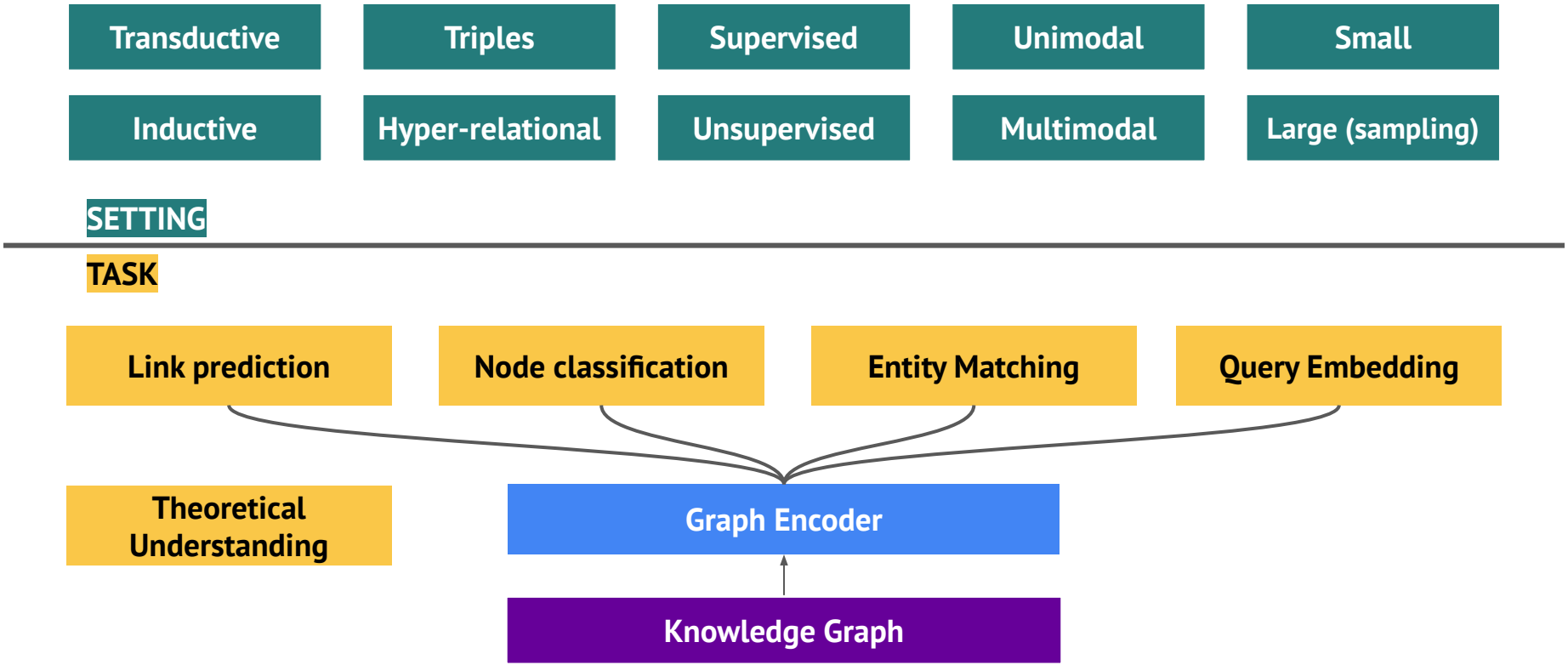
Сегодня

1. Introduction
2. Представление знаний в графах - RDF & RDFS & OWL
3. Хранение знаний в графах - SPARQL & Graph Databases
4. Однородность знаний - RDF* & Wikidata & SHACL & ShEx
5. Интеграция данных в графы знаний - Semantic Data Integration
6. Введение в теорию графов - Graph Theory Intro
7. Векторные представления графов - Knowledge Graph Embeddings
8. **Машинное обучение на графах - Graph Neural Networks & KGs**
9. Некоторые применения - Question Answering & Query Embedding

Сегодня

- Message Passing Intuition
  - GCN, GAT, MPNN
- Relational GCN (R-GCN)
- Compositional GCN (CompGCN)
- Inductive Learning
  - Out-of-Sample Representation Learning
  - Textual Descriptions as Features
  - Inductive Representation Learning

Big Picture in $\mathbb{R}^5$



Knowledge Graph Embeddings

Lecture 7

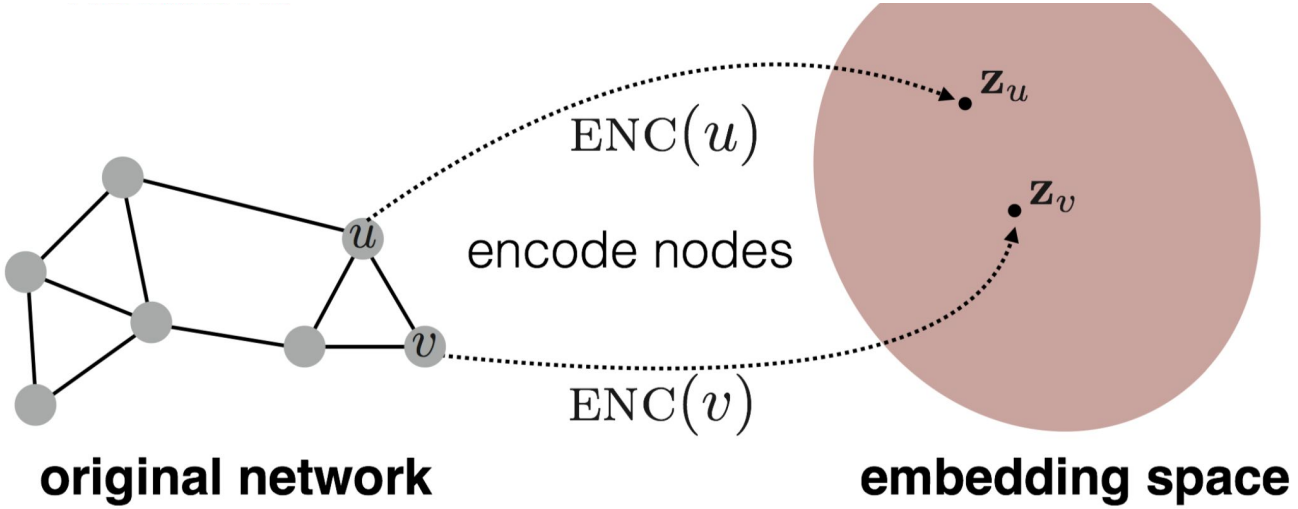
Tensor Factorization

Translation

Neural Networks

Graph Neural Nets

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

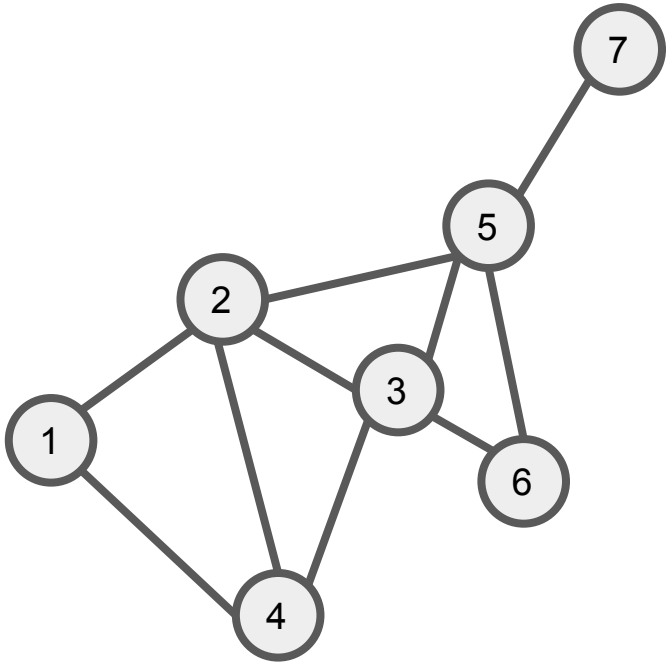


Source: Stanford CS224w, http://web.stanford.edu/class/cs224w/

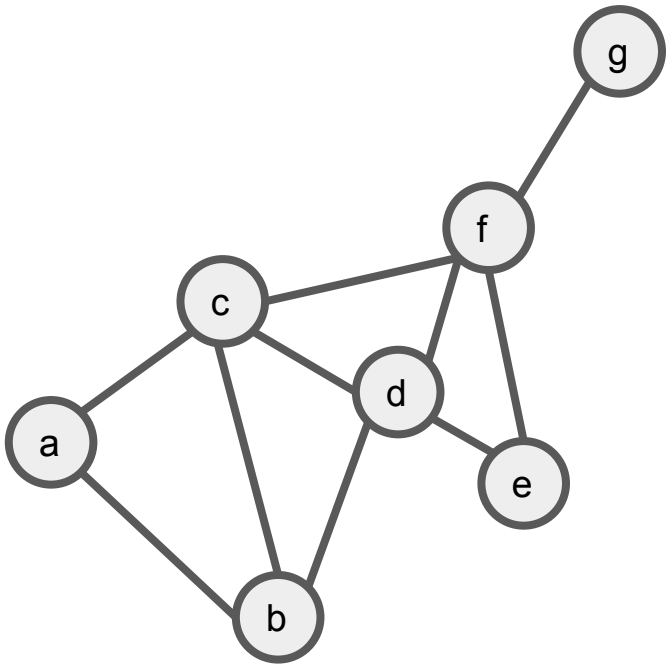
Message Passing

Классический граф

- Ненаправленный
- Нет типов связей



Message Passing - вход



Что у нас есть:

- **A** - матрица смежности (adjacency matrix)

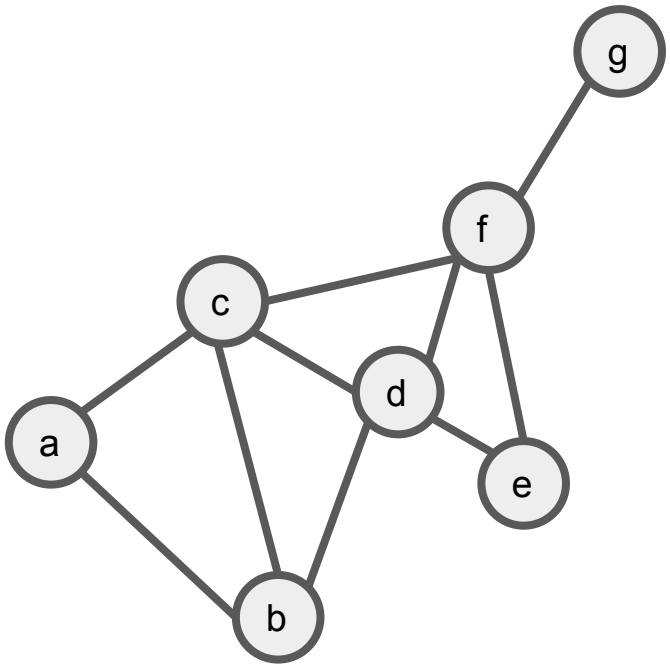
$$\mathbf{A} : N \times N$$

Для экономии памяти:

- Разреженная (sparse) A
- Лист граней (edge index)

```
[[ a, a, b, b, c, c, d, d, f, f],
 [ b, c, c, d, d, f, f, e, e, g]]
```

Message Passing - вход



Что у нас есть:

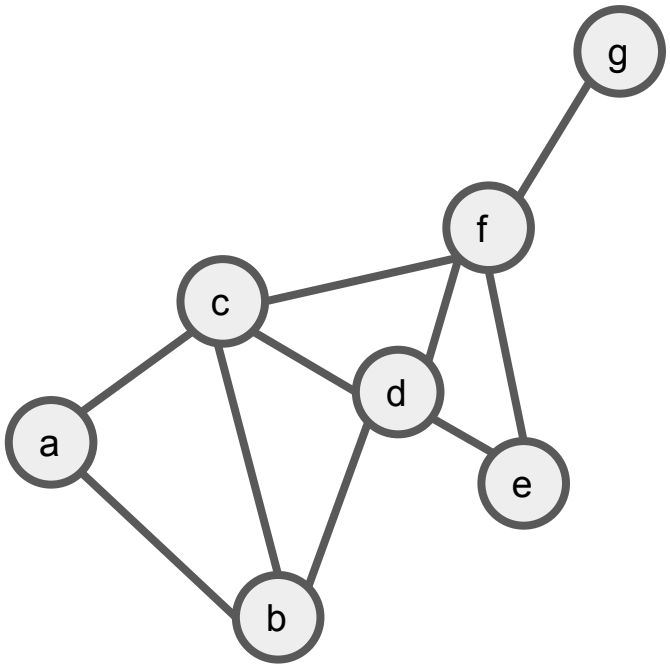
- **A** - матрица смежности (adjacency matrix)

$$\mathbf{A} : N \times N$$

- **X** - признаки вершин (заданы или обучаемые)

$$\mathbf{X} \in \mathbb{R}^{N \times d}$$

Message Passing - выход



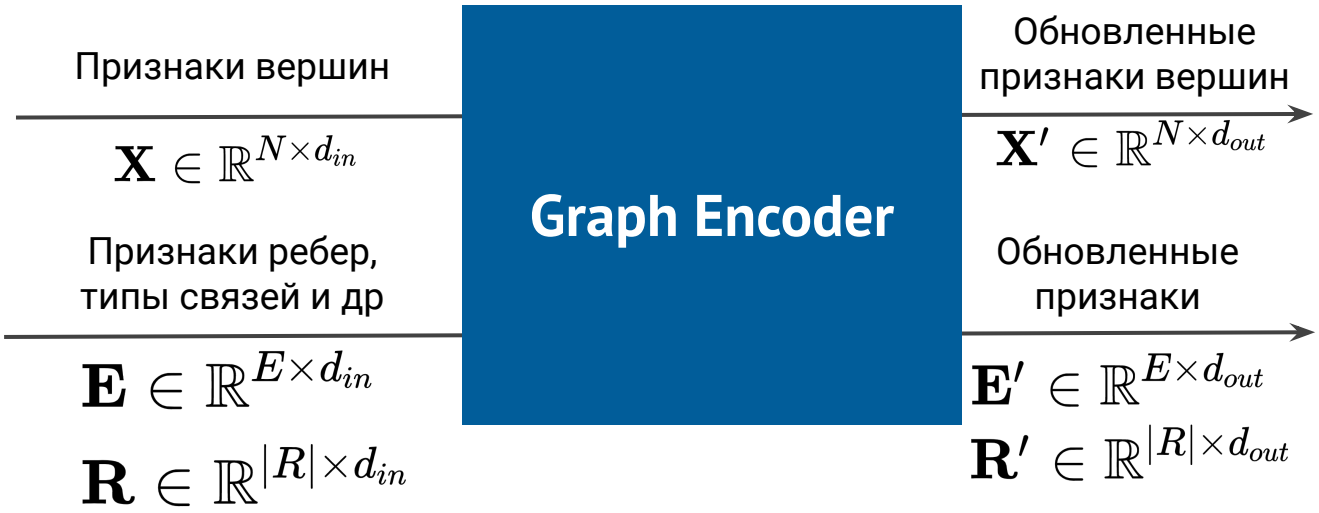
Вход:

- **A, X, [R, ...]** $\mathbf{X} \in \mathbb{R}^{N \times d_{in}}$

Выход:
Обновленные представления узлов [типов связей и др]

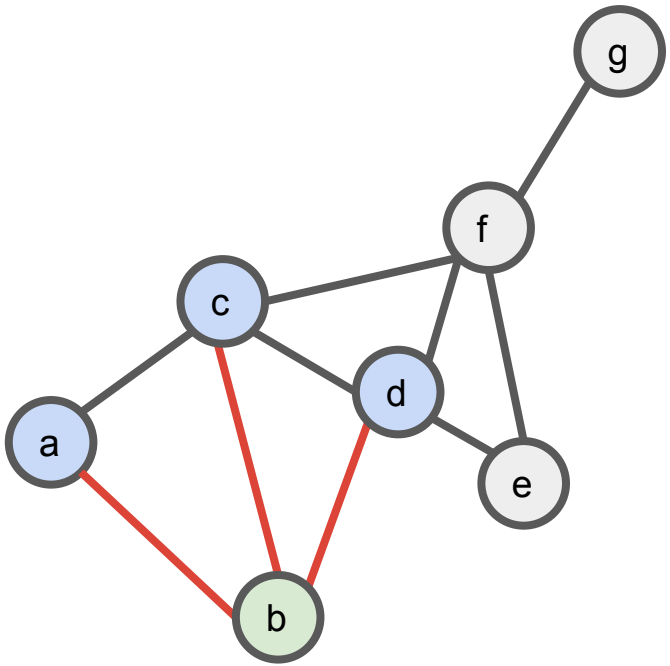
- A, **X'** , [**R'**, ...] $\mathbf{X}' \in \mathbb{R}^{N \times d_{out}}$

Message Passing - выход



- Node Classification
- Link Prediction
- Query Embedding
- Other tasks

Message Passing = Neighborhood Aggregation



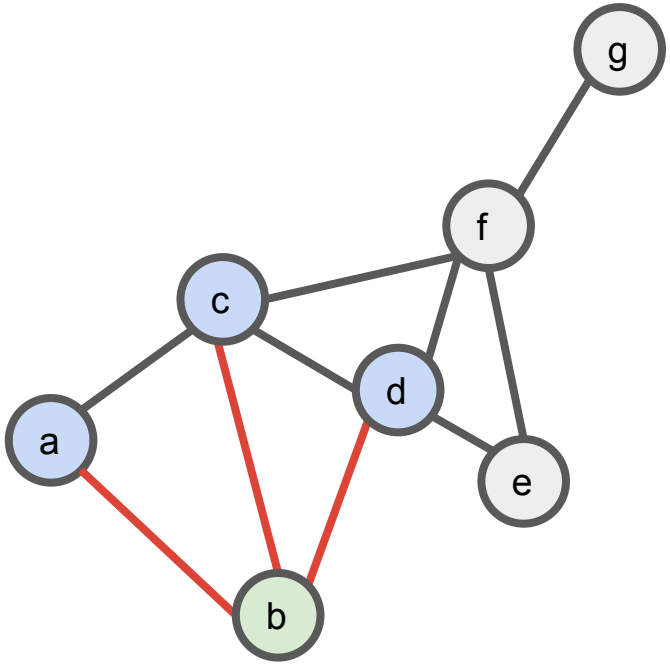
Новое представление узла b получается как **функция** от предыдущего представления **x(b)** и представления соседей **X(*N(b)*)**

$$\mathbf{h}_b = \phi(\mathbf{x}_b, \mathbf{X}_{\mathcal{N}_b})$$

Представление соседей получается путем **агрегации** их представлений

$$\mathbf{X}_{\mathcal{N}_b} = \psi(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d,)$$

Message Passing: Aggregate + Update



1. AGGREGATE
   Построение сообщения **m**

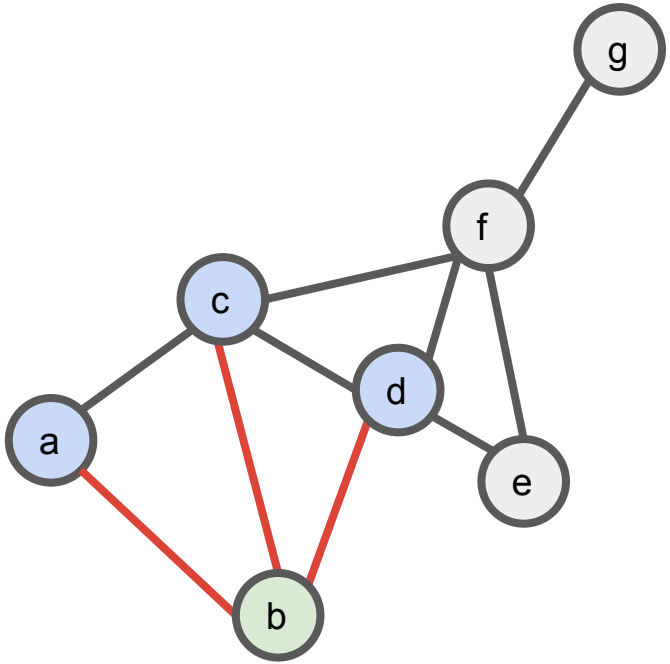
$$\mathbf{m}_{\mathcal{N}(u)}^{(k)} = \text{AGGREGATE}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})$$

2. UPDATE
   Обновление узла **u**

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)})$$

William L. Hamilton. Graph Representation Learning. Morgan & Claypool, 2020

Message Passing: Aggregate



1. AGGREGATE
   Построение сообщения **m**

$$\mathbf{m}_{\mathcal{N}(u)}^{(k)} = \text{AGGREGATE}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})$$

Permutation invariance агрегаторы

Инвариантные к перестановкам

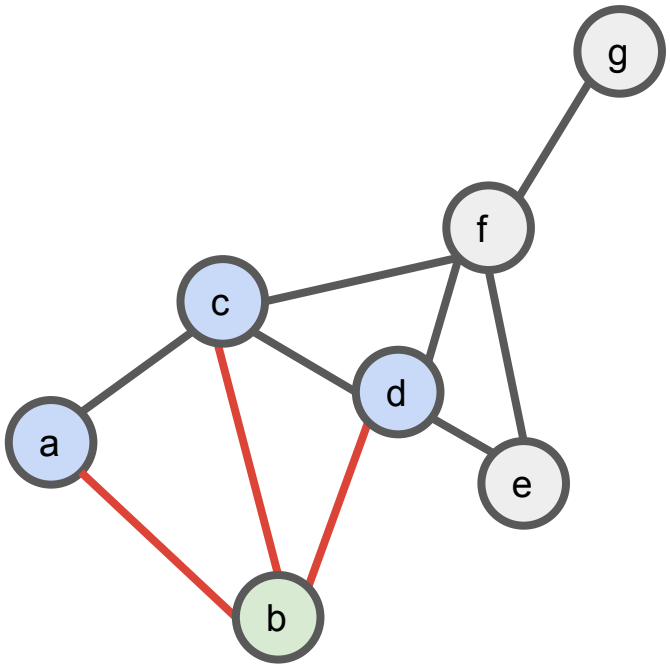
$$\mathbf{m}_{\mathcal{N}(u)} = \bigoplus_{v \in \mathcal{N}(u)} \psi(\mathbf{x}_v)$$

Например, **суммирование**

$$\mathbf{m}_{\mathcal{N}(u)} = \mathbf{W}_{\text{neigh}} \sum_{v \in \mathcal{N}(u)} \mathbf{x}_u$$

William L. Hamilton. Graph Representation Learning. Morgan & Claypool, 2020

Message Passing: Update



2. UPDATE
   Обновление узла **u**

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)})$$

или

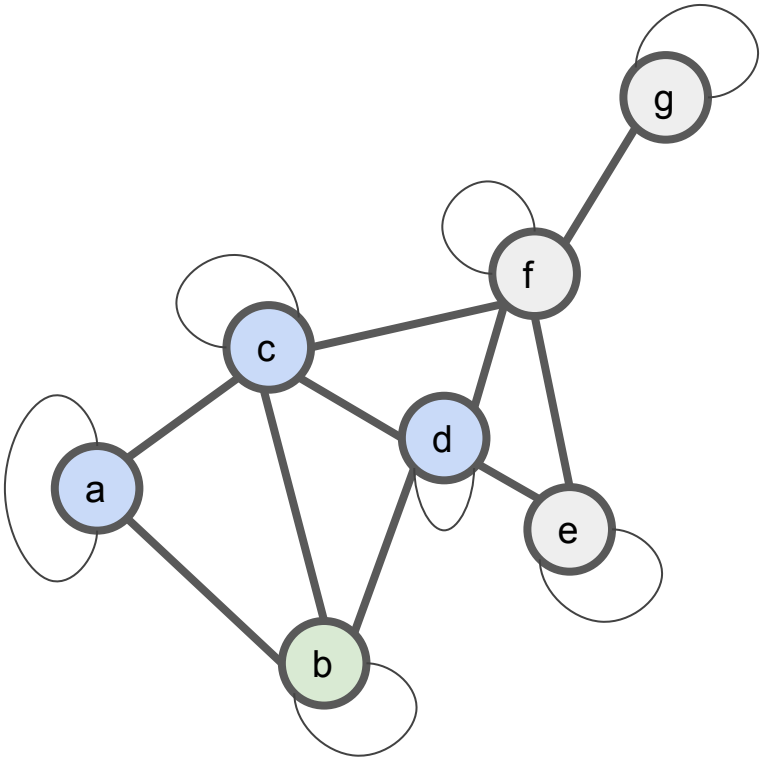
$$\mathbf{h}_u^{(k+1)} = \phi(\mathbf{h}_u^{(k)}, \bigoplus_{v \in \mathcal{N}(u)} \mathbf{h}_v)$$

Нужна **нелинейная** функция
(sigmoid, tanh, ReLU, etc)

$$\mathbf{h}_u^{(k+1)} = \sigma(\mathbf{W}_{\text{self}} \mathbf{h}_u^{(k)} + \mathbf{W}_{\text{neigh}} \mathbf{m}_{\mathcal{N}(u)})$$

William L. Hamilton. Graph Representation Learning. Morgan & Claypool, 2020

Message Passing: Матричная запись



Простейший способ с добавлением self-loops

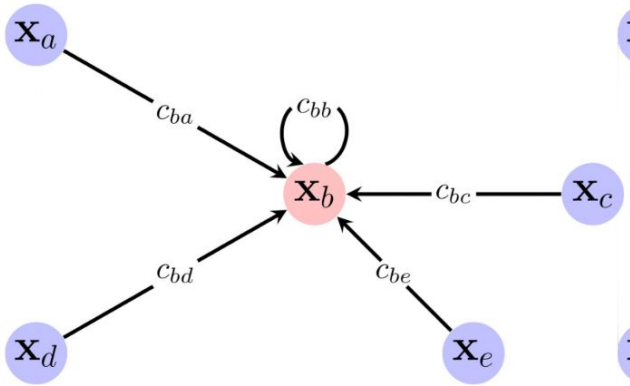
$$\mathbf{h}_u^{(k+1)} = \sigma(\mathbf{W}\mathbf{h}_u^{(k)} + \mathbf{W}\mathbf{m}_{\mathcal{N}(u)})$$

эквивалентен

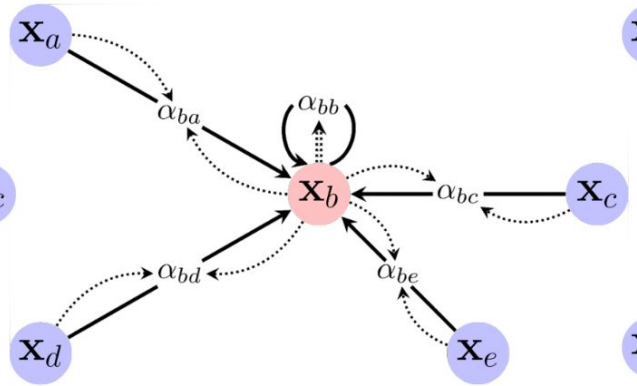
$$\mathbf{H}^{(k+1)} = \sigma((\mathbf{A}+\mathbf{I})\mathbf{H}^{(k)}\mathbf{W}^{(k)})$$
$$= \sigma(\tilde{\mathbf{A}}\mathbf{H}^{(k)}\mathbf{W}^{(k)})$$

William L. Hamilton. Graph Representation Learning. Morgan & Claypool, 2020

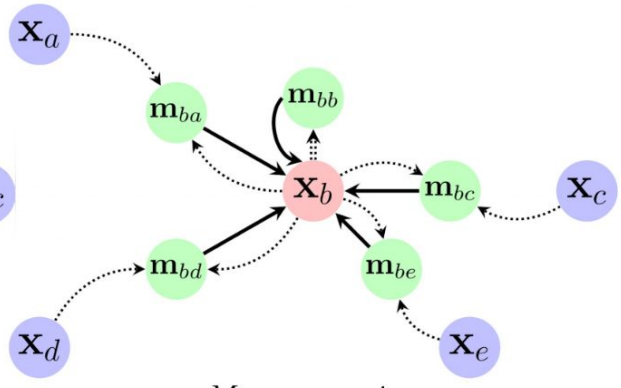
Message Passing Flavours



Convolutional



Attentional



Message-passing

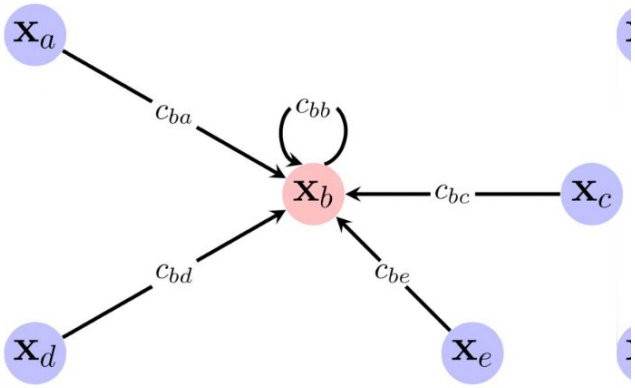
$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Michael M. Bronstein, Joan Bruna, Taco Cohen, Petar Veličković. Geometric Deep Learning. 2021

Graph Convolutional Nets (GCN)



Convolutional

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$

- Фиксированные веса c_{ij} агрегации представлений соседей (константа зависит от модели)
- Стандартный GCN взвешивает в зависимости от степени вершины:

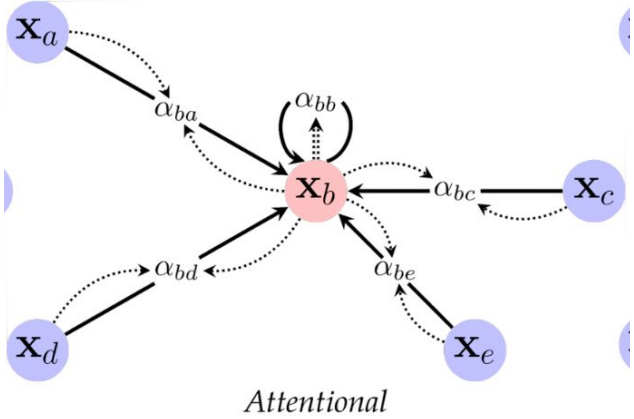
$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right)$$

- Матричная форма:

$$H = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} XW$$

Kipf and Welling. Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017
William L. Hamilton. Graph Representation Learning. Morgan & Claypool, 2020

Graph Attention Nets (GAT)



Attentional

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$

- Обучаемые веса c_{ij} агрегации представлений соседей (через **attention**)

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

- Attention coefficients:

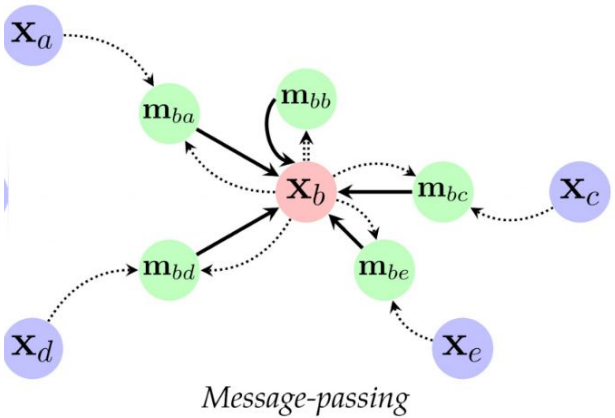
$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

- Multi-head форма:

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

Veličković et al. Graph Attention Networks. ICLR 2018

Message Passing Nets (MPNN)



Message-passing

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

- Наиболее общий вариант - включить представления граней (edge features) в построение message

$$\mathbf{h}_v^{k+1} = \mathrm{UPD}\left(\mathbf{h}_v^k, \underset{u \in \mathcal{N}(v)}{\mathrm{AGGR}}\, \phi(\mathbf{h}_v^k, \mathbf{h}_u^k, \mathbf{e}_{vu})\right)$$

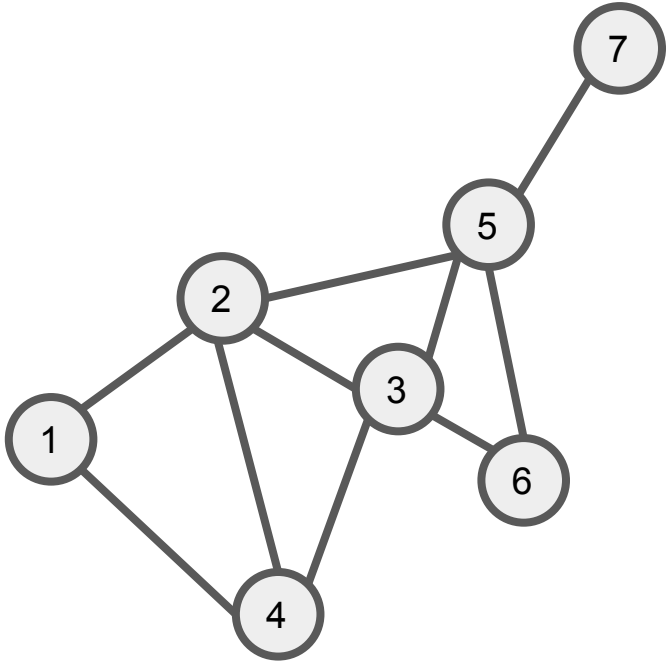
- Edge features могут быть заданы или обучаемы (например, обучаемые типы связей)

Gilmer et al. Neural Message Passing for Quantum Chemistry. ICML 2017

Почему просто не взять GCN / GAT?

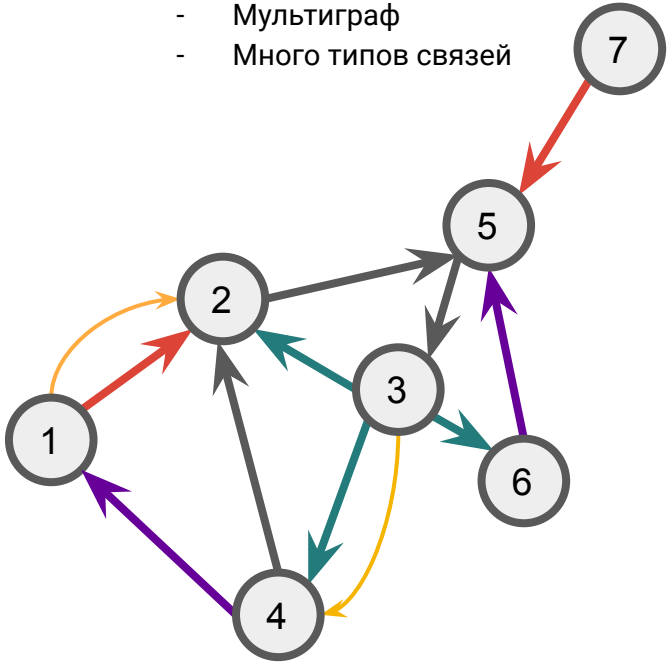
Классический граф

- Ненаправленный
- Нет типов связей



Наш клиент

- Направленный
- Мультиграф
- Много типов связей



Multi-relational GNN Encoders

$$\mathbf{h}_v^{(k)} = f\left(\sum_{u \in \mathcal{N}(v)} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}\right)$$

Vanilla GCN: без типов связей

[1] Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018
[2] Vashishth et al. Composition-Based Multi-Relational Graph Convolutional Networks. ICLR 2020

Multi-relational GNN Encoders

$$\mathbf{h}_v^{(k)} = f\left(\sum_{u \in \mathcal{N}(v)} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}\right)$$

Vanilla GCN: без типов связей

$$\mathbf{h}_v^{(k)} = f\left(\sum_{(u,r) \in \mathcal{N}(v)} \mathbf{W}_r^{(k)} \mathbf{h}_u^{(k-1)}\right)$$

R-GCN [1]: каждый тип связи параметризуется собственной **матрицей** W_r

[1] Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018
[2] Vashishth et al. Composition-Based Multi-Relational Graph Convolutional Networks. ICLR 2020

Multi-relational GNN Encoders

$$\mathbf{h}_v^{(k)} = f\left(\sum_{u \in \mathcal{N}(v)} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}\right)$$

Vanilla GCN: без типов связей

$$\mathbf{h}_v^{(k)} = f\left(\sum_{(u,r) \in \mathcal{N}(v)} \mathbf{W}_r^{(k)} \mathbf{h}_u^{(k-1)}\right)$$

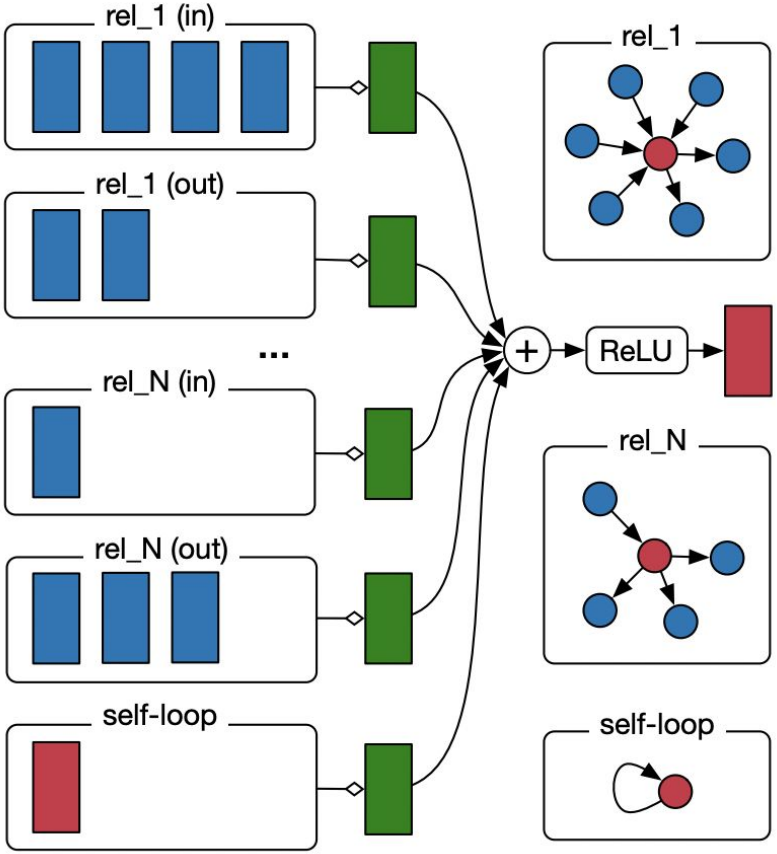
R-GCN [1]: каждый тип связи параметризуется собственной **матрицей** W_r

$$\boldsymbol{h}_v = f\left(\sum_{(u,r) \in \mathcal{N}(v)} \boldsymbol{W}_{\lambda(r)} \phi(\boldsymbol{x}_u, \boldsymbol{z}_r)\right)$$

CompGCN [2]: каждый тип связи параметризуется **вектором** z_r +
композиция (узел, связь) +
3 весовые матрицы W: input/output/self-loop

[1] Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018
[2] Vashishth et al. Composition-Based Multi-Relational Graph Convolutional Networks. ICLR 2020

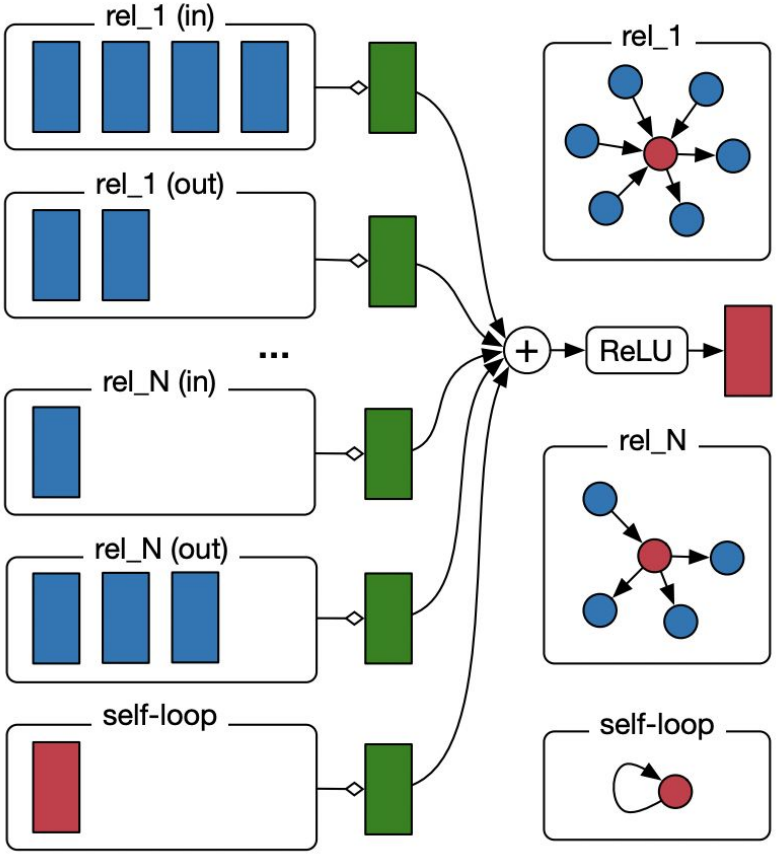
Relational GCN (R-GCN) - Message and Update



$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right)$$

[1] Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018
[2] William Hamilton. Graph Representation Learning. Morgan & Claypool, 2020

Relational GCN (R-GCN) - Message and Update



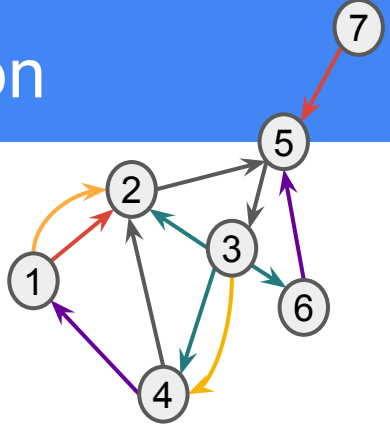
$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right)$$

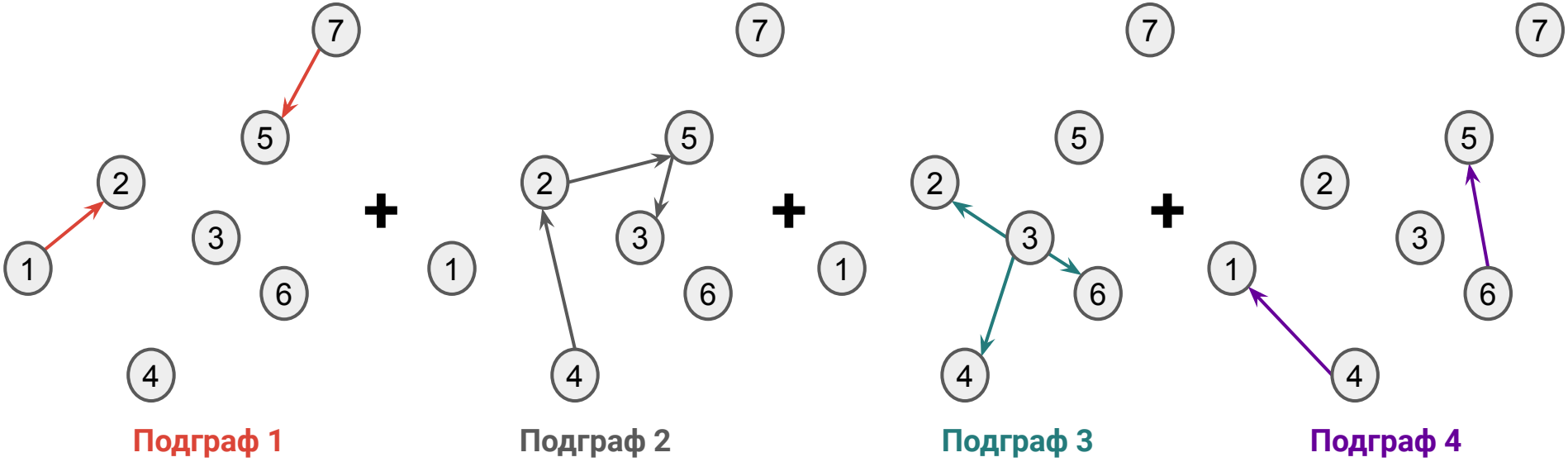
- Каждый тип связи параметризуется собственной **матрицей W_r**
- Нормализация через степень вершины (GCN-style)
- Базисная декомпозиция весов предикатов для экономии параметров

[1] Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018
[2] William Hamilton. Graph Representation Learning. Morgan & Claypool, 2020

Relational GCN (R-GCN) - Graph Decomposition

- Граф из *K* предикатов разбивается на *K* **подграфов** с собственной матрицей смежности (adj matrix)
- Применяем message passing к каждому подграфу отдельно

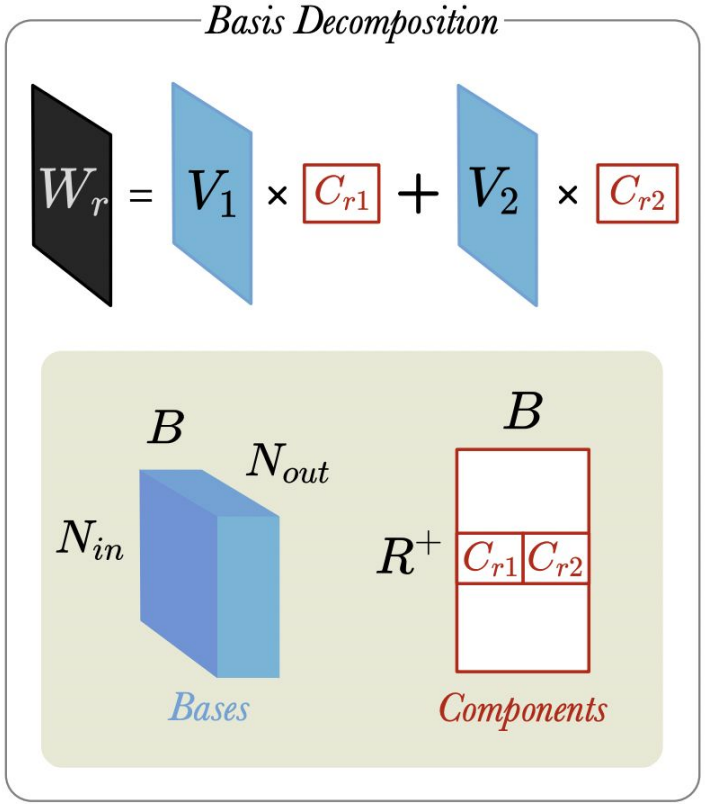




Подграф 1 **Подграф 2** **Подграф 3** **Подграф 4**

[1] Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018
[2] William Hamilton. Graph Representation Learning. Morgan & Claypool, 2020

Relational GCN (R-GCN) Basis Decomposition



Вместо K матриц будем хранить b базисных матриц и получать остальные как линейную комбинацию базисных

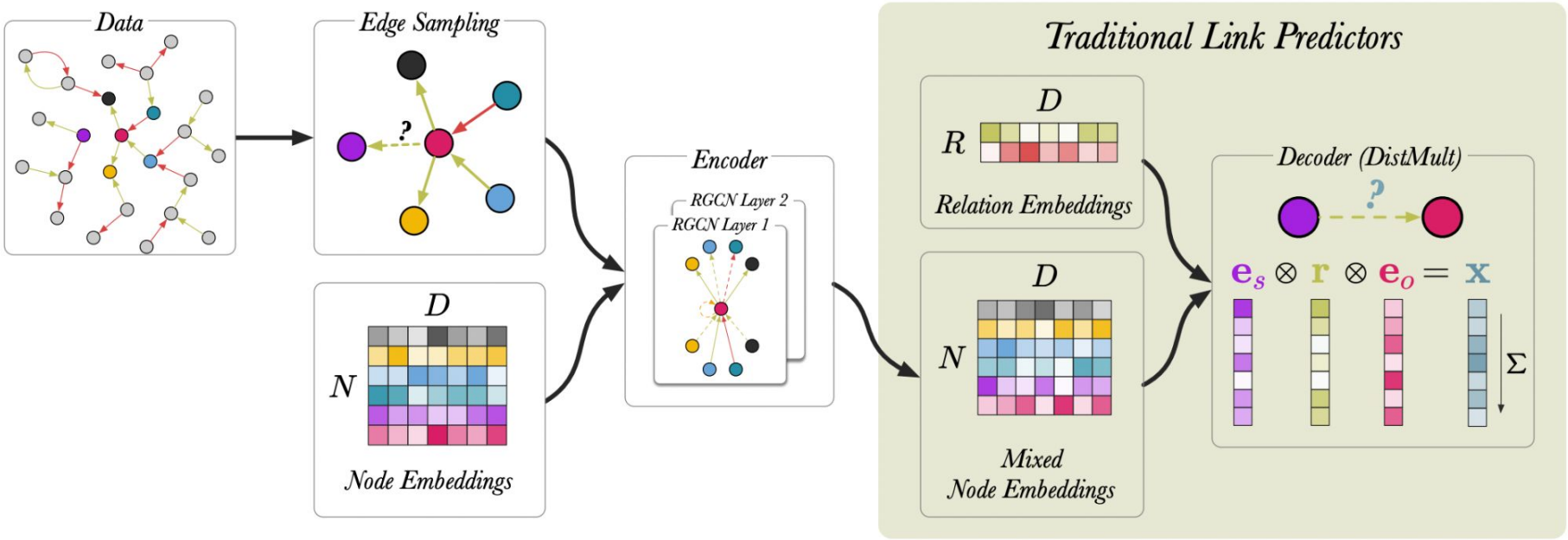
$$\mathbf{W}_\tau = \sum_{i=1}^{b} \alpha_{i,\tau} \mathbf{B}_i.$$

Тогда message function **m**:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{\tau \in \mathcal{R}} \sum_{v \in \mathcal{N}_\tau(u)} \frac{\boldsymbol{\alpha}_\tau \times_1 \mathcal{B} \times_2 \mathbf{h}_v}{f_n(\mathcal{N}(u), \mathcal{N}(v))}$$

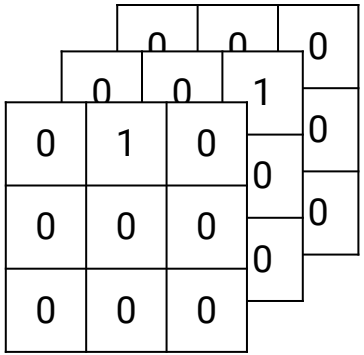
[1] Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018
[2] William Hamilton. Graph Representation Learning. Morgan & Claypool, 2020
[3] Thanapalasingam et al. Relational Graph Convolutional Networks: A Closer Look. arxiv:2021

Relational GCN (R-GCN) Decoders



[1] Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018
[2] William Hamilton. Graph Representation Learning. Morgan & Claypool, 2020
[3] Thanapalasingam et al. Relational Graph Convolutional Networks: A Closer Look. arxiv:2021

Relational GCN (R-GCN) Input Formats

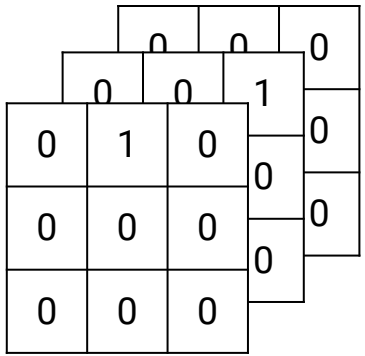


$$\mathcal{T} : \mathbb{R}^{|E| \times |E| \times |R|}$$

- |R| квадратных матриц смежности
- Эффективно только с использованием разреженных матриц (sparse matrices) (scipy.sparse, torch.sparse, etc)

[1] Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018

Relational GCN (R-GCN) Input Formats



$$\mathcal{T} : \mathbb{R}^{|E| \times |E| \times |R|}$$

- |R| квадратных матриц смежности
- Эффективно только с использованием разреженных матриц (sparse matrices) (scipy.sparse, torch.sparse, etc)

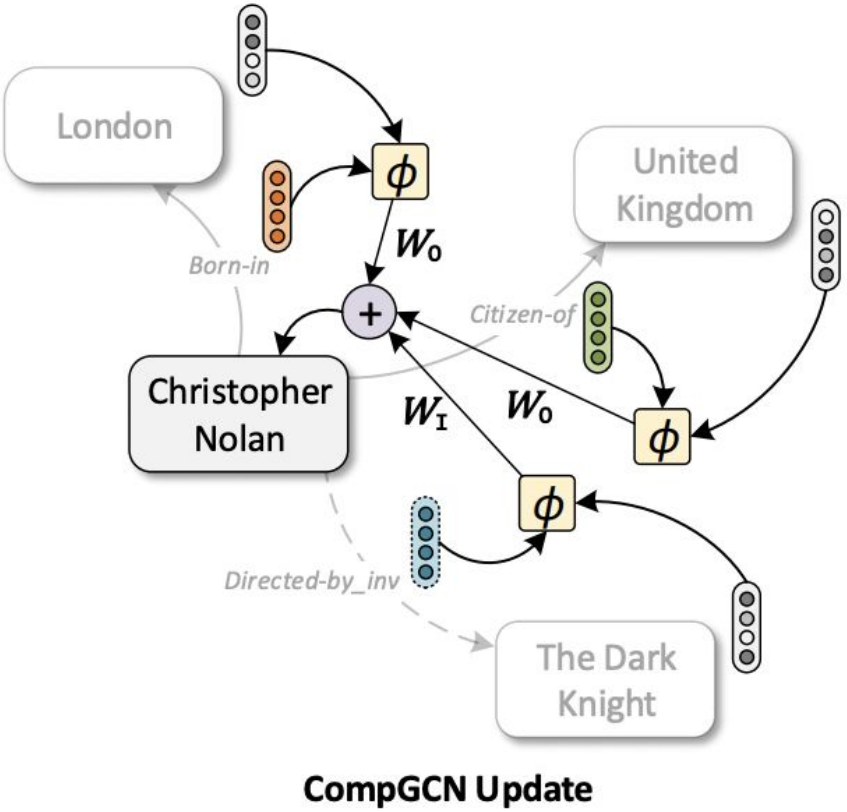
```
r1: [[1, 3, 3, 2, ...], # <- source
     [2, 2, 1, 4, ...]] # <- target

r2: [2, num_edges_r2]
r3: [2, num_edges_r3]
...
r_n: [2, num_edges_rn]
```

- |R| матриц в COO формате (edge index)

[1] Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018

Compositional GCN (CompGCN)



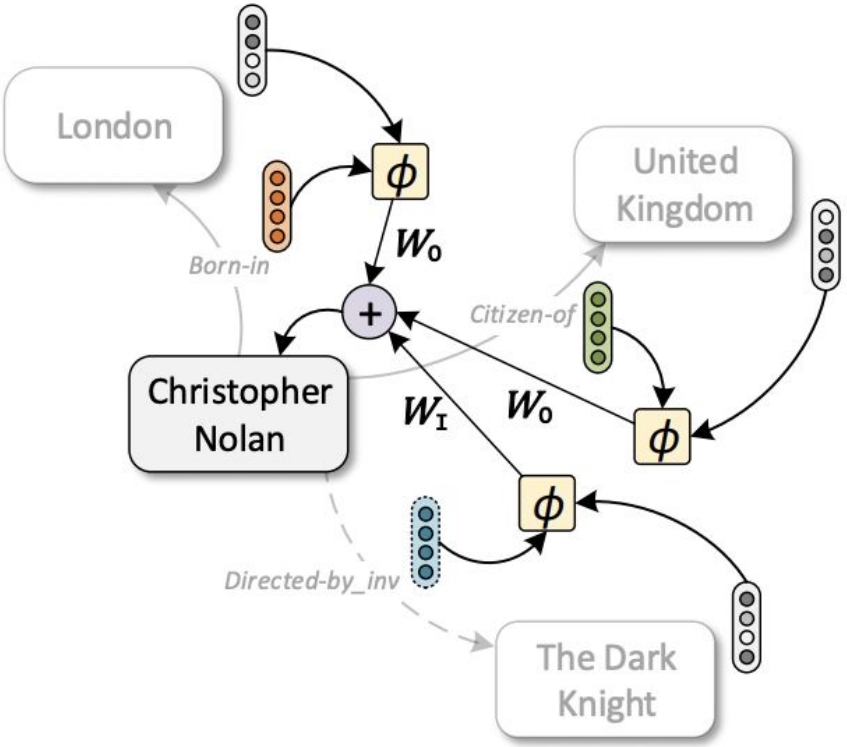
CompGCN Update

$$h_v = f\left(\sum_{(u,r)\in\mathcal{N}(v)} W_{\lambda(r)}\phi(x_u, z_r)\right)$$

- Каждый тип связи параметризуется **вектором z_r**
  - Значительное уменьшение количества параметров
- Сообщения строятся как **композиция** вектора узла и типа связи (функция ϕ)
  - Любая (e,r) функция
  - TransE
  - RotatE, etc

Vashishth et al. Composition-Based Multi-Relational Graph Convolutional Networks. ICLR 2020

Compositional GCN (CompGCN)



CompGCN Update

$$h_v = f\left(\sum_{(u,r)\in\mathcal{N}(v)} W_{\lambda(r)}\phi(x_u, z_r)\right)$$

- Только 3 весовые матрицы **Wλ**
  - Оригинальные связи (direct)
  - Инверсные связи (inverse)
  - Self-loops

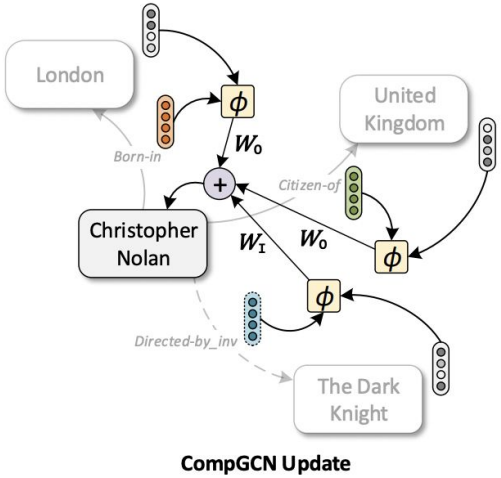
$$W_{\text{dir}(r)} = \begin{cases} W_O, & r \in \mathcal{R} \\ W_I, & r \in \mathcal{R}_{inv} \\ W_S, & r = \top \textit{ (self-loop)} \end{cases}$$

Типы связей на следующем слое:

$$h_r = W_{\text{rel}} z_r$$

Vashishth et al. Composition-Based Multi-Relational Graph Convolutional Networks. ICLR 2020

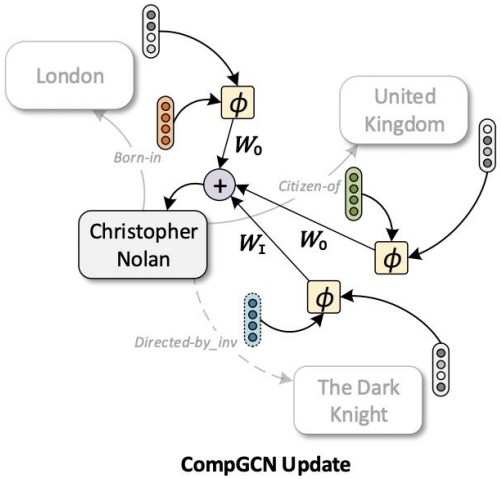
Compositional GCN (CompGCN): COO Input




```
[[1, 3, 3, 2, ...], # <- source
 [2, 2, 1, 4, ...], # <- target
 [0, 1, 1, 2, ...]] # <- edge types
```


Vashishth et al. Composition-Based Multi-Relational Graph Convolutional Networks. ICLR 2020

Compositional GCN (CompGCN): COO Input




```
[[1, 3, 3, 2, ...], # <- source
 [2, 2, 1, 4, ...], # <- target
 [0, 1, 1, 2, ...]] # <- edge types
```

```
[[1, 3, 3, 2, ..., 2, 2, 1, 4, ..., 1, 2, 3, 4, ...  ], # <- source
 [2, 2, 1, 4, ..., 1, 3, 3, 2, ..., 1, 2, 3, 4, ...  ], # <- target
 [0, 1, 1, 2, ..., k+1, k+3, k+3, k+2, ..., sl, sl, sl, sl, ... ]] # <- edge types
```

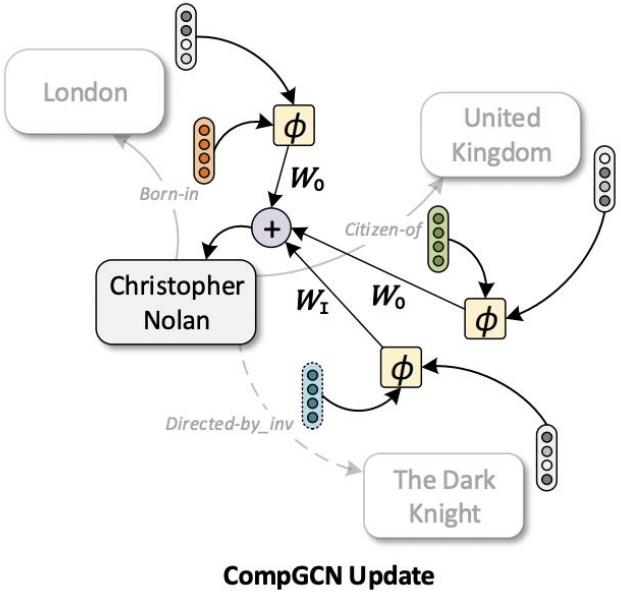
Direct edges

Inverse edges, **new edge types**

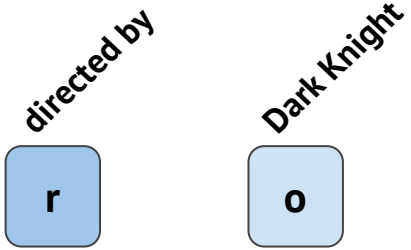
Self-loops + special edge type

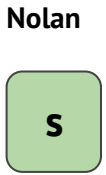
Vashishth et al. Composition-Based Multi-Relational Graph Convolutional Networks. ICLR 2020

Compositional GCN (CompGCN): COO Input



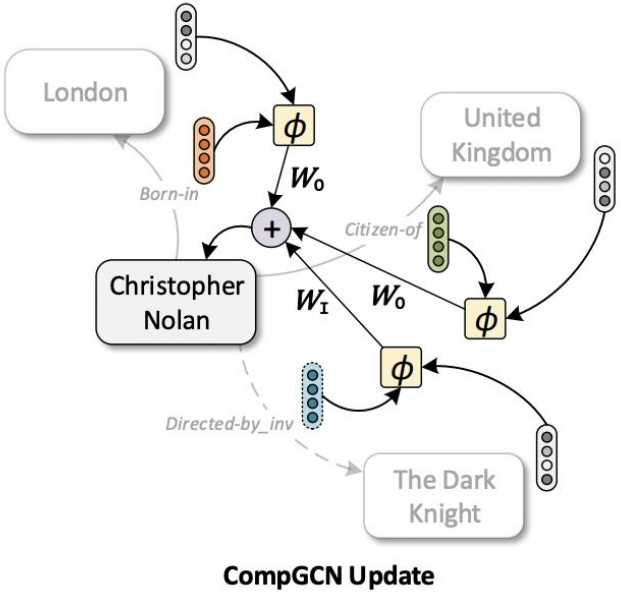
CompGCN Update



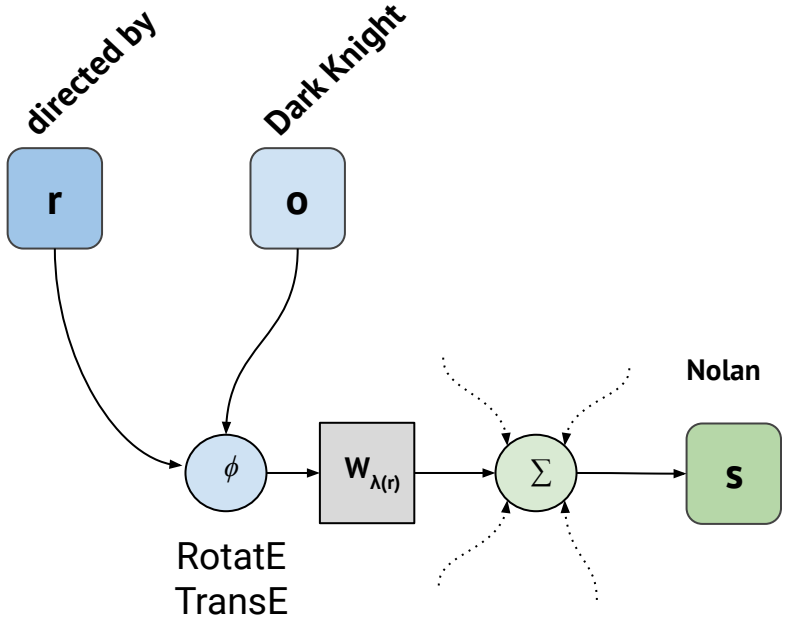


Vashishth et al. Composition-Based Multi-Relational Graph Convolutional Networks. ICLR 2020

Compositional GCN (CompGCN): COO Input



CompGCN Update



RotatE
TransE
DistMult
etc

Vashishth et al. Composition-Based Multi-Relational Graph Convolutional Networks. ICLR 2020

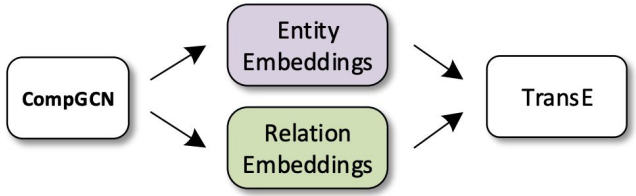
Compositional GCN (CompGCN) - Performance

	FB15k-237					WN18RR				
	MRR	MR	H@10	H@3	H@1	MRR	MR	H@10	H@3	H@1
TransE (Bordes et al., 2013)	.294	357	.465	-	-	.226	3384	.501	-	-
DistMult (Yang et al., 2014)	.241	254	.419	.263	.155	.43	5110	.49	.44	.39
ComplEx (Trouillon et al., 2016)	.247	339	.428	.275	.158	.44	5261	.51	.46	.41
R-GCN (Schlichtkrull et al., 2017)	.248	-	.417		.151	-	-	-		-
KBGAN (Cai & Wang, 2018)	.278	-	.458		-	.214	-	.472	-	-
ConvE (Dettmers et al., 2018)	.325	244	.501	.356	.237	.43	4187	.52	.44	.40
ConvKB (Nguyen et al., 2018)	.243	311	.421	.371	.155	.249	**3324**	.524	.417	.057
SACN (Shang et al., 2019)	.35	-	**.54**	**.39**	.26	.47	-	.54	.48	.43
HypER (Balažević et al., 2019)	.341	250	.520	.376	.252	.465	5798	.522	.477	.436
RotatE (Sun et al., 2019)	.338	**177**	.533	.375	.241	.476	3340	**.571**	.492	.428
ConvR (Jiang et al., 2019)	.350	-	.528	.385	.261	.475	-	.537	.489	**.443**
VR-GCN (Ye et al., 2019)	.248	-	.432	.272	.159	-	-	-	-	-
CompGCN (Proposed Method)	**.355**	197	**.535**	**.390**	**.264**	**.479**	3533	.546	**.494**	**.443**

Vashishth et al. Composition-Based Multi-Relational Graph Convolutional Networks. ICLR 2020

Compositional GCN (CompGCN) - Performance

Как декодер можно взять любую scoring function из shallow моделей (Lecture 7)

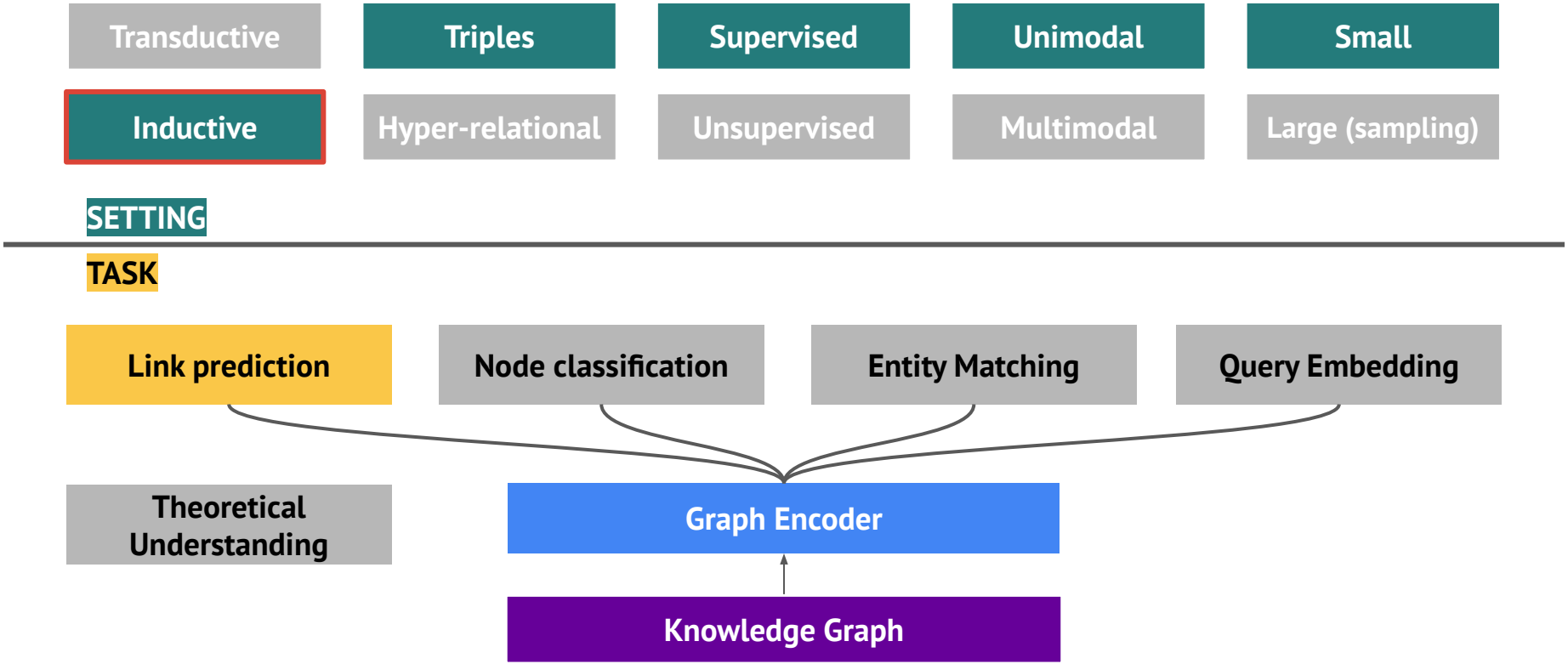


Scoring Function (=X) →	TransE			DistMult			ConvE		
Methods ↓	MRR	MR	H@10	MRR	MR	H@10	MRR	MR	H@10
X	0.294	357	0.465	0.241	354	0.419	0.325	244	0.501
X + D-GCN	0.299	351	0.469	0.321	225	0.497	0.344	200	0.524
X + R-GCN	0.281	325	0.443	0.324	230	0.499	0.342	197	0.524
X + W-GCN	0.267	1520	0.444	0.324	229	0.504	0.344	201	0.525
X + CompGCN (Sub)	0.335	**194**	0.514	0.336	231	0.513	0.352	199	0.530
X + CompGCN (Mult)	**0.337**	233	0.515	**0.338**	**200**	**0.518**	0.353	216	0.532
X + CompGCN (Corr)	0.336	214	**0.518**	0.335	227	0.514	**0.355**	197	**0.535**
X + CompGCN ($\mathcal{B} = 50$)	0.330	203	0.502	0.333	210	0.512	0.350	**193**	0.530

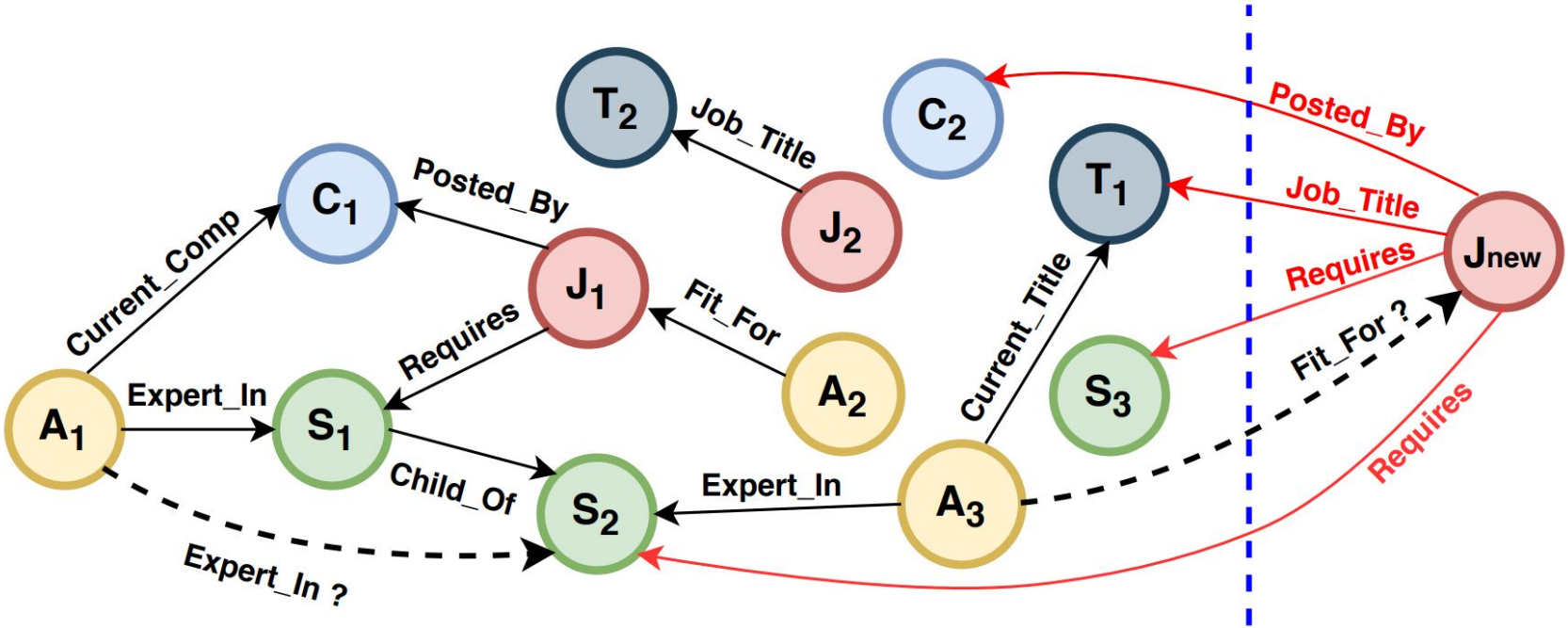
Vashishth et al. Composition-Based Multi-Relational Graph Convolutional Networks. ICLR 2020

Inductive Learning

Inductive Learning



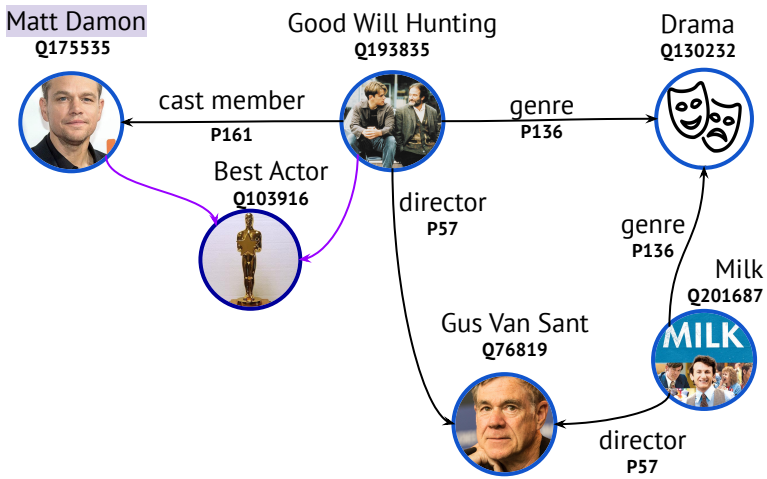
Inductive Learning 1 - Добавление к известному графу



- На инференс приходит новая сущность Jnew, для которой у нас нет эмбеддинга
- Моделирует в т.ч. и темпоральные условия (изменения графа во времени)

Albooyeh et al. Out-of-Sample Representation Learning for Knowledge Graphs. EMNLP 2020

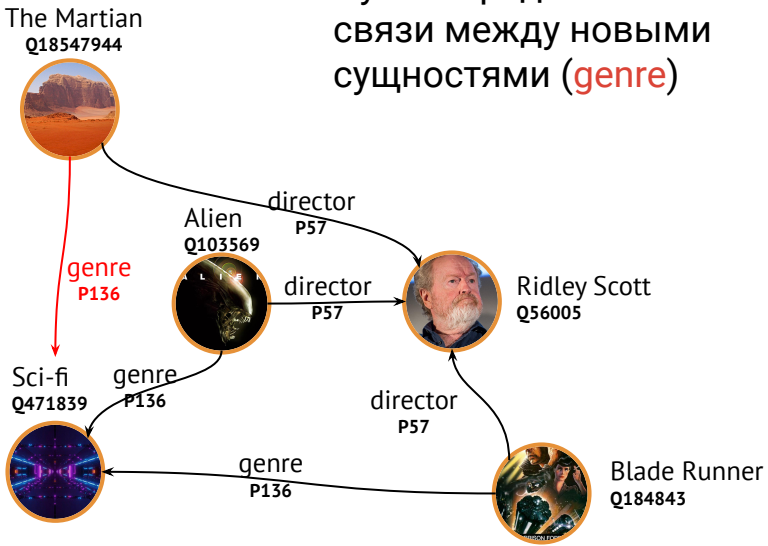
Inductive Learning 2 - Новый инференс граф



Тренировочный граф

Тест (Inference)

- Новые сущности
- Известные предикаты
- Нужно предсказывать связи между новыми сущностями (genre)



Transductive vs Inductive

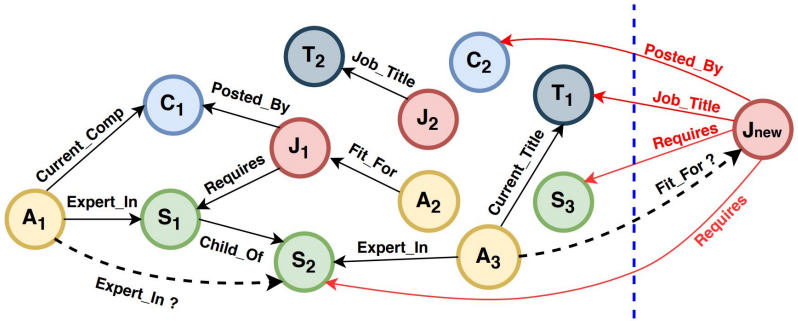
Transductive
(трансдуктивный)

1. Весь граф известен во время тренировки
2. Применимы shallow / decoder-only модели (TransE, RotatE, ConvE, ComplEx, etc) для тренировки
3. Полученная матрица entity embeddings применима на тесте/инференсе

Inductive
(индуктивный)

1. Валидация/тест/инференс - новые сущности или новый граф
   a. Relation types не изменяются
2. Shallow / decoder-only модели не могут выучить эмбеддинг неизвестного узла (unseen node)
   a. **Как построить node feature?**
3. GNNs могут работать в индуктивном сценарии
   a. Используя дополнительные node features
   b. Используя представления подграфов или путей

Inductive Learning 1 - Добавление к известному графу



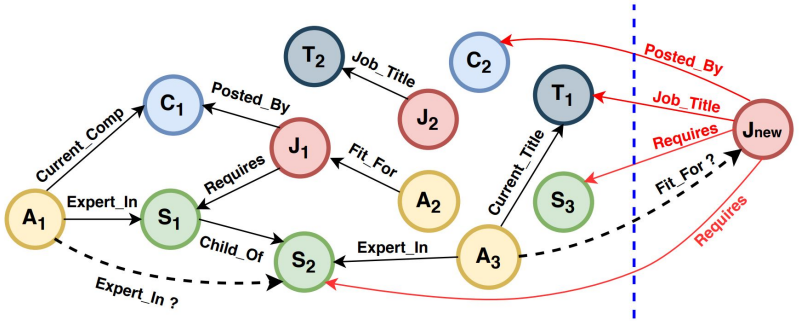
DistMult

$$z_v = \frac{1}{|\mathcal{G}_v|}\left(\sum_{(v,r,u)\in\mathcal{G}_v} z_r \odot z_u + \sum_{(u,r,v)\in\mathcal{G}_v} z_r \odot z_u \right)$$

- Идея: эмбеддинг новой сущности как среднее от представлений соседей

Albooyeh et al. Out-of-Sample Representation Learning for Knowledge Graphs. EMNLP 2020

Inductive Learning 1 - Добавление к известному графу



DistMult

$$\boldsymbol{z}_{\mathsf{v}} = \frac{1}{|\mathcal{G}_{\mathsf{v}}|}\Big(\sum_{(\mathsf{v},\mathsf{r},\mathsf{u})\in\mathcal{G}_{\mathsf{v}}} \boldsymbol{z}_{\mathsf{r}} \odot \boldsymbol{z}_{\mathsf{u}} + \sum_{(\mathsf{u},\mathsf{r},\mathsf{v})\in\mathcal{G}_{\mathsf{v}}} \boldsymbol{z}_{\mathsf{r}} \odot \boldsymbol{z}_{\mathsf{u}} \Big)$$

- Идея: эмбеддинг новой сущности как среднее от представлений соседей
- Новый способ тренировки для имитации “новых” сущностей

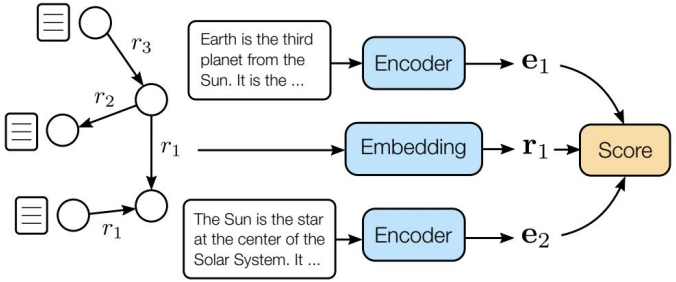
Algorithm 2 Out-of-Sample Training (one epoch)

Inputs n : negative ratio, $\mathcal{L}$: loss function, ψ : see Section 3.1

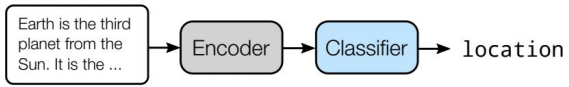
for $batch = 1$ **to** $numBatches$ **do**
  $triples, labels \leftarrow$ `nextBatch` $(batch, n)$
  $scores \leftarrow []$
  for $(\mathsf{v}, \mathsf{r}, \mathsf{u})$ **in** $triples$ **do**
    $\boldsymbol{z}_{\mathsf{r}} \leftarrow \texttt{lookup}(\mathsf{r}, \boldsymbol{Z}_{\mathsf{r}})$
    $rand \leftarrow \texttt{random}()$
    if $rand < \frac{\psi}{2}$ **then**
      $\boldsymbol{z}_{\mathsf{v}} \leftarrow \texttt{aggregate}(\mathsf{v}, \boldsymbol{Z}_{rels}, \boldsymbol{Z}_{ent})$
      $\boldsymbol{z}_{\mathsf{u}} \leftarrow \texttt{lookup}(\mathsf{u}, \boldsymbol{Z}_{ent})$
    else if $\frac{\psi}{2} < rand < \psi$ **then**
      $\boldsymbol{z}_{\mathsf{v}} \leftarrow \texttt{lookup}(\mathsf{v}, \boldsymbol{Z}_{ent})$
      $\boldsymbol{z}_{\mathsf{u}} \leftarrow \texttt{aggregate}(\mathsf{u}, \boldsymbol{Z}_{rel}, \boldsymbol{Z}_{ent})$
    else
      $\boldsymbol{z}_{\mathsf{v}} \leftarrow \texttt{lookup}(\mathsf{v}, \boldsymbol{Z}_{ent})$
      $\boldsymbol{z}_{\mathsf{u}} \leftarrow \texttt{lookup}(\mathsf{u}, \boldsymbol{Z}_{ent})$
    end if
    $scores.\texttt{append}(\phi(\boldsymbol{z}_{\mathsf{v}}, \boldsymbol{z}_{\mathsf{r}}, \boldsymbol{z}_{\mathsf{u}}))$
  end for
  `updateParams`$(\mathcal{L}, scores, labels)$
end for

Albooyeh et al. Out-of-Sample Representation Learning for Knowledge Graphs. EMNLP 2020

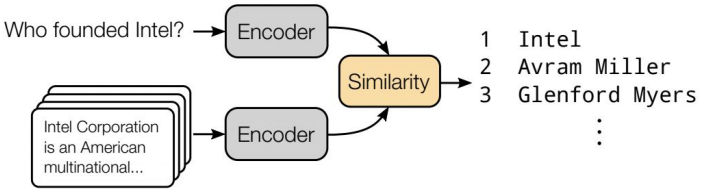
Inductive Learning 2 - Text Descriptions as Features



Link prediction



Entity classification



Information retrieval

- Тренировать эмбеддинги на train graph бессмысленно, **НО** у всех сущностей часто есть текстовое описание (Wikidata, DBpedia, Wikipedia)
- Будем использовать языковую модель (BERT, etc) как универсальный энкодер (featurizer) этих описаний
- Полученные эмбеддинги можно использовать в link prediction, entity classification, IR

Daza et al. Inductive Entity Representations from Text via Link Prediction. WWW 2021

Inductive Learning 2 - Text Descriptions as Features

Стандартная схема: encoder + decoder (task-specific)

- Например, BERT + TransE

	WN18RR				FB15k-237				Wikidata5M			
Method	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
GloVe-BOW	0.170	0.055	0.215	0.405	0.172	0.099	0.188	0.316	0.343	0.092	0.531	0.756
BE-BOW	0.180	0.045	0.244	0.450	0.173	0.103	0.184	0.316	0.362	0.082	0.586	0.798
GloVe-DKRL	0.115	0.031	0.141	0.282	0.112	0.062	0.111	0.211	0.282	0.077	0.403	0.660
BE-DKRL	0.139	0.048	0.169	0.320	0.144	0.084	0.151	0.263	0.322	0.097	0.474	0.720
KEPLER	–	–	–	–	–	–	–	–	0.402	0.222	0.514	0.730
BLP-TransE	**0.285**	0.135	**0.361**	**0.580**	**0.195**	**0.113**	**0.213**	**0.363**	0.478	0.241	0.660	0.871
BLP-DistMult	0.248	0.135	0.288	0.481	0.146	0.076	0.156	0.286	0.472	0.242	0.646	0.869
BLP-ComplEx	0.261	**0.156**	0.297	0.472	0.148	0.081	0.154	0.283	0.489	0.262	**0.664**	**0.877**
BLP-SimplE	0.239	0.144	0.265	0.435	0.144	0.077	0.152	0.274	**0.493**	**0.289**	0.639	0.866

Daza et al. Inductive Entity Representations from Text via Link Prediction. WWW 2021

Inductive Learning 2 - Subgraphs

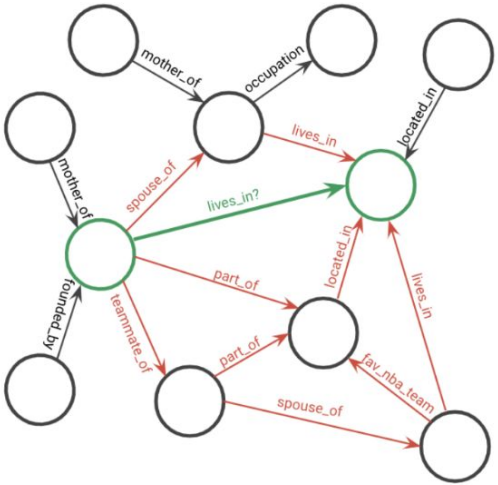
- Текстовых описаний нет
- Откуда брать node embeddings?

Teru et al. Inductive Relation Prediction by Subgraph Reasoning. ICML 2020

Inductive Learning 2 - Subgraphs

- Текстовых описаний нет
- Откуда брать node embeddings?

- Тренируем relation embeddings
- Получаем node features из структуры подграфов (sample subgraphs)



1. Sample the enclosing sub-graph around the link to be predicted (target link).

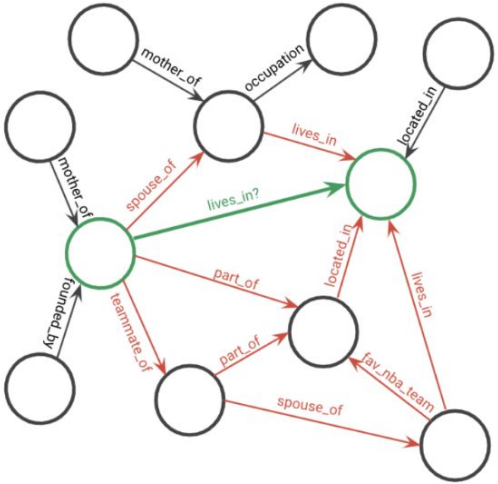
Сэмплируем подграф вокруг двух заданных узлов

Teru et al. Inductive Relation Prediction by Subgraph Reasoning. ICML 2020

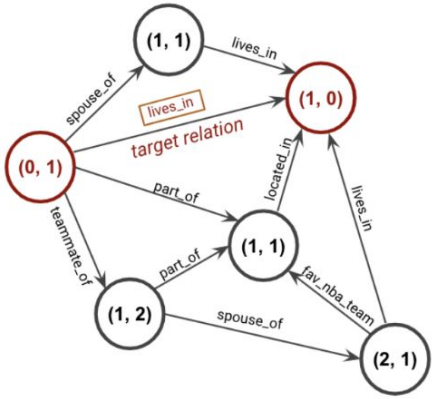
Inductive Learning 2 - Subgraphs

- Текстовых описаний нет
- Откуда брать node embeddings?

- Тренируем relation embeddings
- Получаем node features из структуры подграфов (sample subgraphs)



1. Sample the enclosing sub-graph around the link to be predicted (target link).



2. Label the nodes w.r.t the target nodes to identify their structural role. Uniquely labels target nodes to mark them for the model.

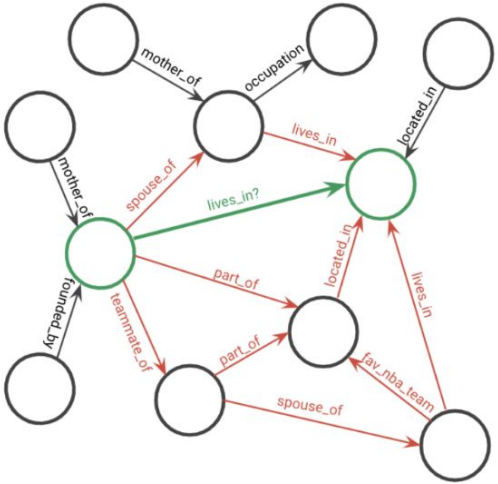
Получаем кратчайшие пути через алгоритм Дейкстры

Teru et al. Inductive Relation Prediction by Subgraph Reasoning. ICML 2020

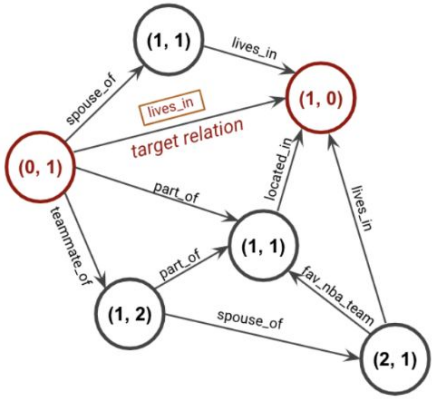
Inductive Learning 2 - Subgraphs

- Текстовых описаний нет
- Откуда брать node embeddings?

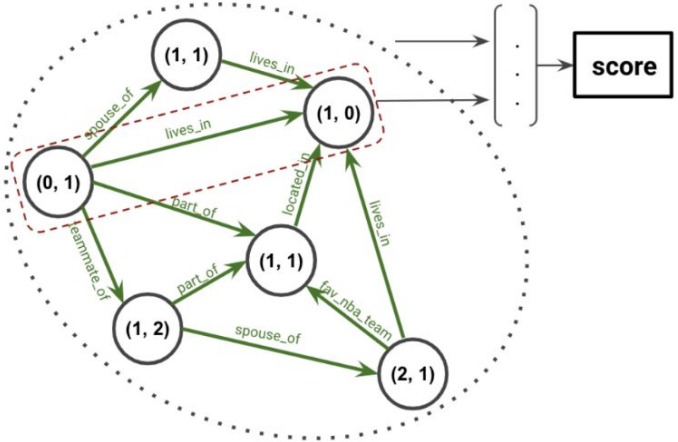
- Тренируем relation embeddings
- Получаем node features из структуры подграфов (sample subgraphs)



1. Sample the enclosing sub-graph around the link to be predicted (target link).



2. Label the nodes w.r.t the target nodes to identify their structural role. Uniquely labels target nodes to mark them for the model.



3. Pass messages across the (sub-)graph to predict a score indicating how strongly the structure around the target link supports its logical plausibility.

Teru et al. Inductive Relation Prediction by Subgraph Reasoning. ICML 2020

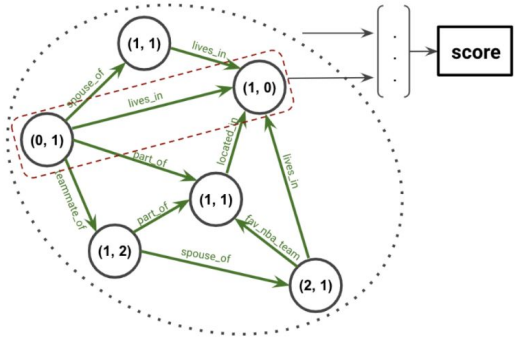
Inductive Learning 2 - Subgraphs - GraIL

1. **Энкодер**: edge-aware R-GCN + attention

Inspired by the multi-relational R-GCN (Schlichtkrull et al., 2017) and edge attention, we define our AGGREGATE function as

$$\mathbf{a}_t^k = \sum_{r=1}^{R} \sum_{s \in \mathcal{N}_r(t)} \alpha_{rr_tst}^k \mathbf{W}_r^k \mathbf{h}_s^{k-1},$$

$$\mathbf{s} = \text{ReLU}\left(\mathbf{A}_1^k [\mathbf{h}_s^{k-1} \oplus \mathbf{h}_t^{k-1} \oplus \mathbf{e}_r^a \oplus \mathbf{e}_{r_t}^a] + \mathbf{b}_1^k\right)$$
$$\alpha_{rr_tst}^k = \sigma\left(\mathbf{A}_2^k \mathbf{s} + \mathbf{b}_2^k\right).$$



3. Pass messages across the (sub-)graph to predict a score indicating how strongly the structure around the target link supports its logical plausibility.

Teru et al. Inductive Relation Prediction by Subgraph Reasoning. ICML 2020

Inductive Learning 2 - Subgraphs - GraIL

1. **Энкодер**: edge-aware R-GCN + attention

Inspired by the multi-relational R-GCN (Schlichtkrull et al., 2017) and edge attention, we define our AGGREGATE function as

$$\mathbf{a}_t^k = \sum_{r=1}^{R} \sum_{s \in \mathcal{N}_r(t)} \alpha_{rr_tst}^k \mathbf{W}_r^k \mathbf{h}_s^{k-1},$$

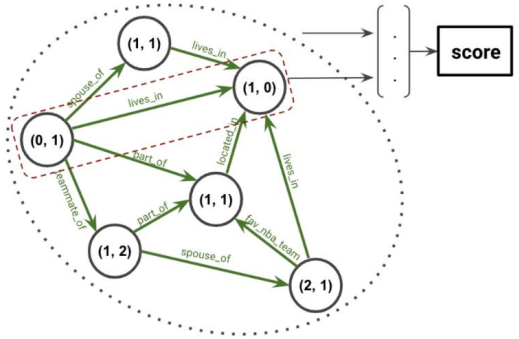
$$\mathbf{s} = \text{ReLU}\left(\mathbf{A}_1^k [\mathbf{h}_s^{k-1} \oplus \mathbf{h}_t^{k-1} \oplus \mathbf{e}_r^a \oplus \mathbf{e}_{r_t}^a] + \mathbf{b}_1^k\right)$$
$$\alpha_{rr_tst}^k = \sigma\left(\mathbf{A}_2^k \mathbf{s} + \mathbf{b}_2^k\right).$$

2. **Pooling**: subgraph vector как среднее от представлений его узлов

$$\mathbf{h}_{\mathcal{G}_{(u,v,r_t)}}^L = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathbf{h}_i^L,$$

3. **Финальный декодер**

$$\text{score}(u, r_t, v) = \mathbf{W}^T [\mathbf{h}_{\mathcal{G}_{(u,v,r_t)}}^L \oplus \mathbf{h}_u^L \oplus \mathbf{h}_v^L \oplus \mathbf{e}_{r_t}].$$



3. Pass messages across the (sub-)graph to predict a score indicating how strongly the structure around the target link supports its logical plausibility.

Teru et al. Inductive Relation Prediction by Subgraph Reasoning. ICML 2020

Inductive Learning 2 - Subgraphs - GraIL

Только relation prediction (не link prediction) из-за высокой вычислительной сложности

Table 1. Inductive results (AUC-PR)

	WN18RR				FB15k-237				NELL-995			
	v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
Neural-LP	86.02	83.78	62.90	82.06	69.64	76.55	73.95	75.74	64.66	83.61	87.58	85.69
DRUM	86.02	84.05	63.20	82.06	69.71	76.44	74.03	76.20	59.86	83.99	87.71	85.94
RuleN	90.26	89.01	76.46	85.75	75.24	88.70	91.24	91.79	84.99	88.40	87.20	80.52
GraIL	**94.32**	**94.18**	**85.80**	**92.72**	**84.69**	**90.57**	**91.68**	**94.46**	**86.05**	**92.62**	**93.34**	**87.50**

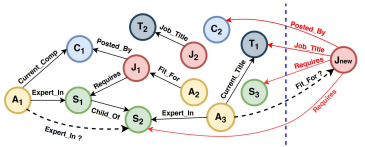
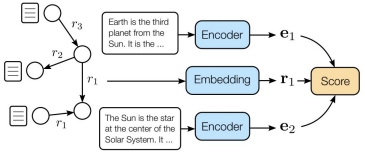
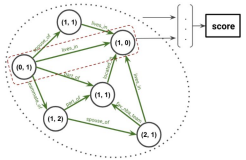
Table 7. Ablation study of the proposed framework (AUC-PR)

	FB (v3)	NELL (v3)
GraIL	**91.68**	**93.34**
GraIL w/o enclosing subgraph	84.25	85.89
GraIL w/o node labeling scheme	82.07	84.46
GraIL w/o attention in GNN	90.27	87.30

Представления подграфов и способ составления node features очень важны

Teru et al. Inductive Relation Prediction by Subgraph Reasoning. ICML 2020

Inductive Learning Summary

Как строим представления неизвестных узлов на инференсе?	Сценарий	Способ
Агрегация тренированных векторов известных нод	Новые узлы (инференс) присоединяются к тренировочному графу (out-of-sample)	
Есть текстовые описания узлов	1. Out-of-sample 2. Полностью новый граф	
Совсем никаких заготовленных features. Только структурные свойства графа	1. Out-of-sample 2. Полностью новый граф	

3. Pass messages across the (sub-)graph to predict a score indicating how strongly the structure around the target link supports its logical plausibility.

В следующей серии

1. Introduction
2. Представление знаний в графах - RDF & RDFS & OWL
3. Хранение знаний в графах - SPARQL & Graph Databases
4. Однородность знаний - RDF* & Wikidata & SHACL & ShEx
5. Интеграция данных в графы знаний - Semantic Data Integration
6. Введение в теорию графов - Graph Theory Intro
7. Векторные представления графов - Knowledge Graph Embeddings
8. Машинное обучение на графах - Graph Neural Networks & KGs
9. **Некоторые применения - Question Answering & Query Embedding**