

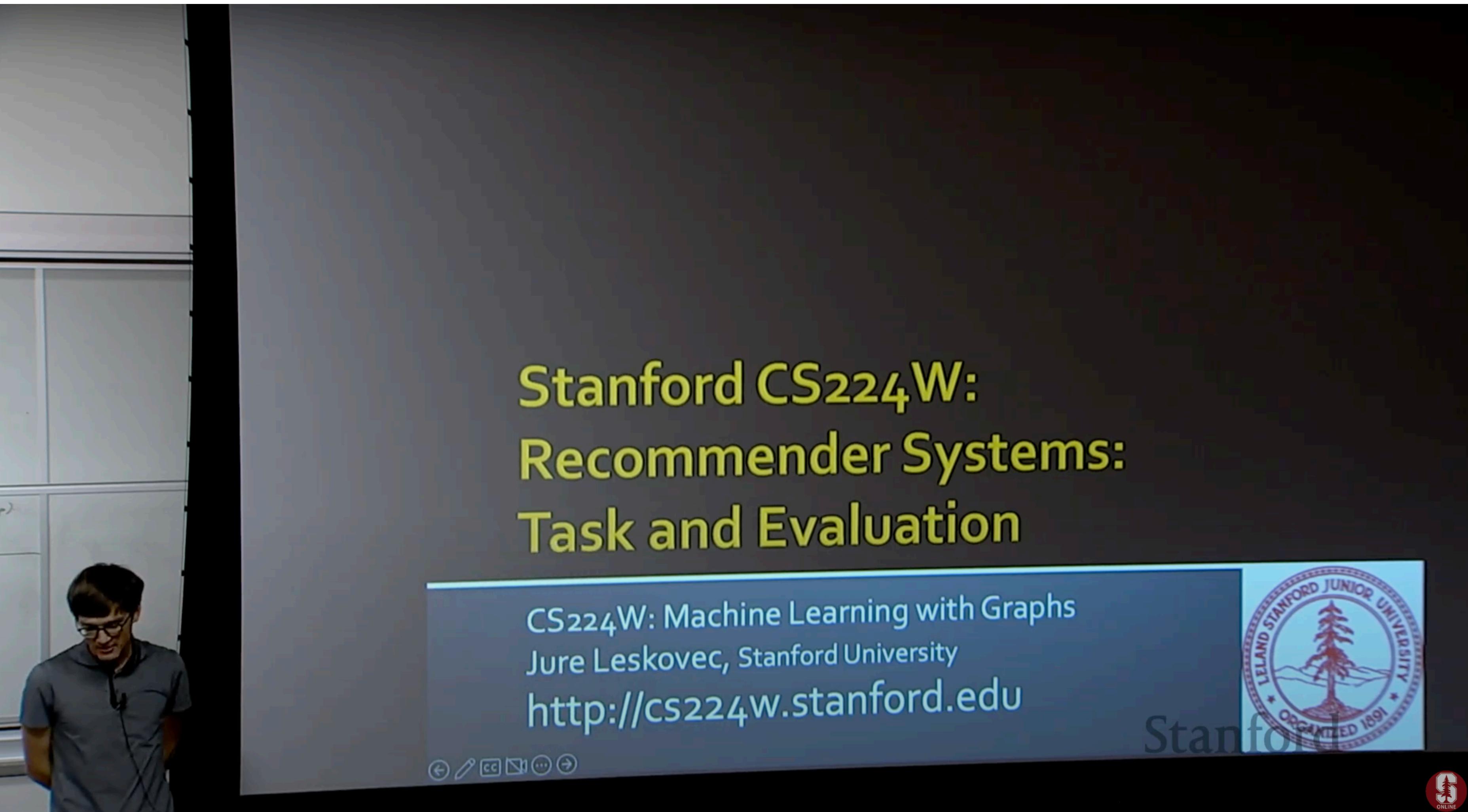


# Графовые модели для рекомендательных систем

ШАД, курс про графы в машинном обучении и курс по  
рекомендательным системам, весна 2025

Кирилл Хрыльченко

# CS224W: Machine Learning with Graphs



The image shows a man with dark hair and glasses, wearing a grey t-shirt, standing to the left of a large projection screen. He appears to be giving a lecture or presentation. The projection screen displays a slide with the following content:

**Stanford CS224W:  
Recommender Systems:  
Task and Evaluation**

CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>

Stanford

LELAND STANFORD JUNIOR UNIVERSITY  
ORGANIZED 1891

CC BY-SA

# Структура лекции

- 1 | Рекомендательные системы
- 2 | Графовая постановка
- 3 | PinSage
- 4 | Продвинутые рекомендательные системы
- 5 | Гетерогенность, TwHIN,  
MultiBiSage, OmniSage

1

>

# Рекомендательные системы

# Рексистемы everywhere

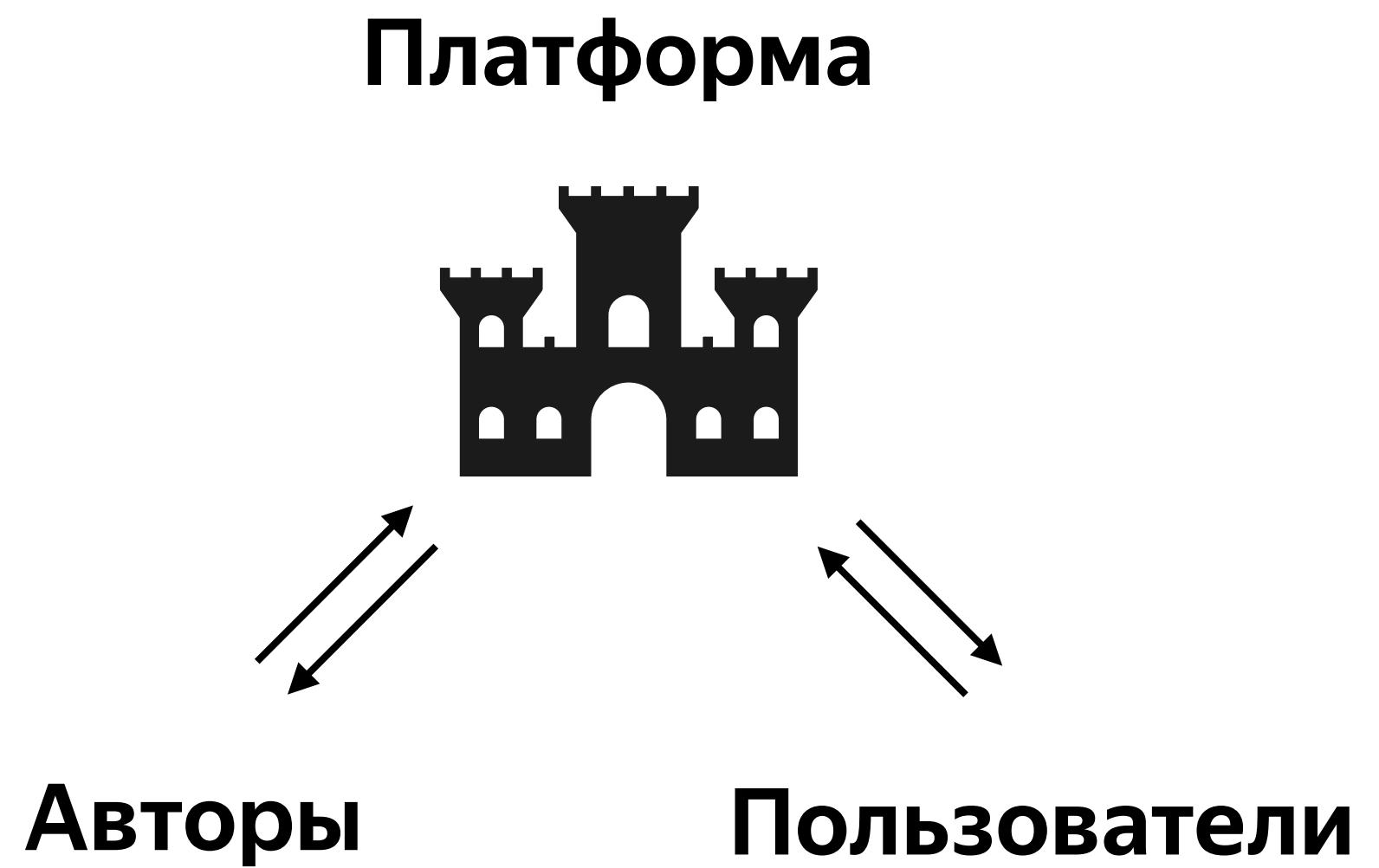
Мы сталкиваемся с рекомендательными системами  
**каждый день:**

- › Слушаем музыку
- › Смотрим видео
- › Читаем новости
- › Совершаем покупки
- › Ищем информацию

# Роль рекомендательных систем

Рекомендательные системы помогают:

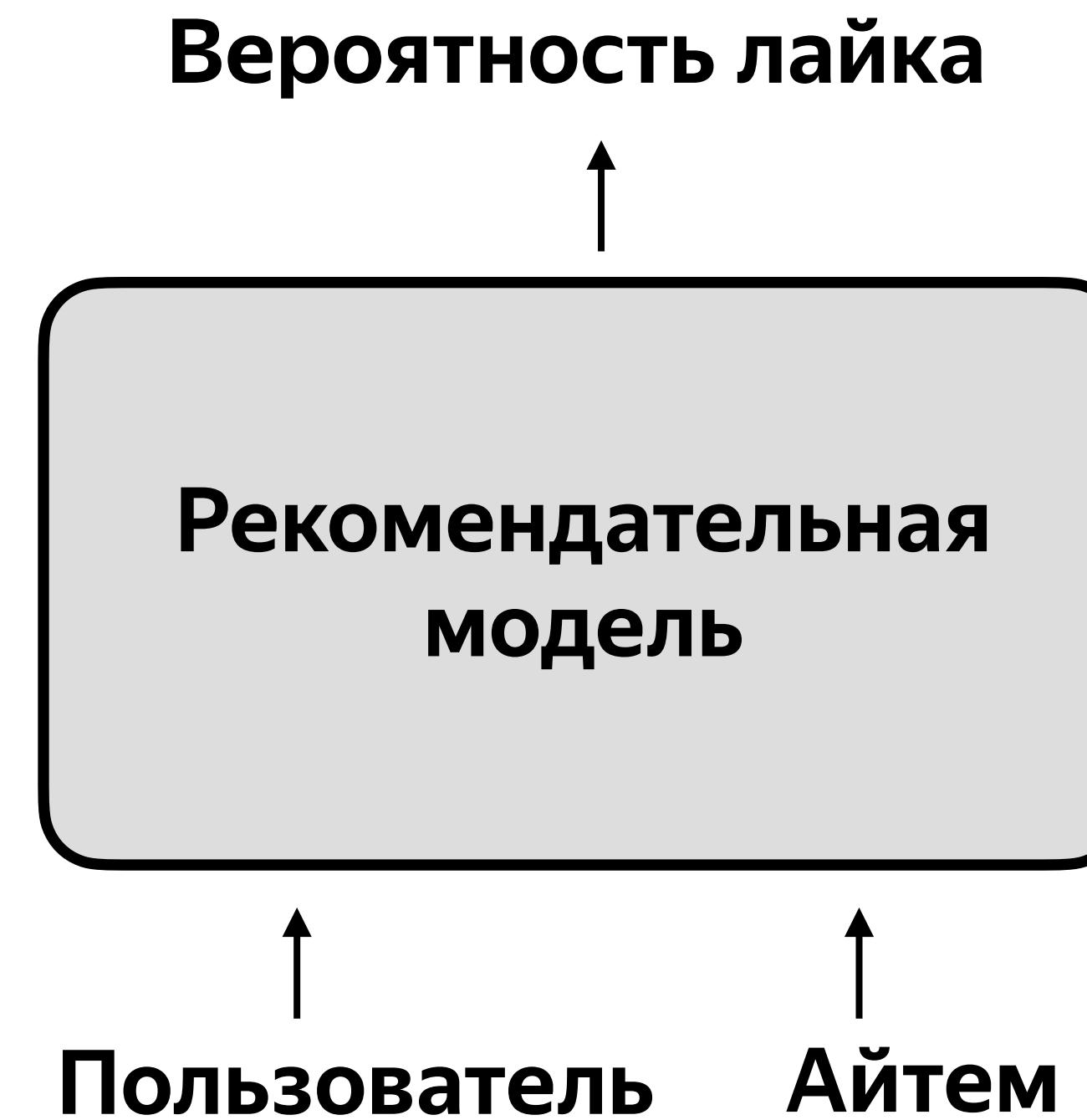
- › Пользователям бороться с информационной перегрузкой
- › Авторам / поставщикам находить аудиторию
- › Платформам зарабатывать



# Рекомендации - это ML

Под капотом рекомендательных систем – **машинное обучение**.

- › Формируем **признаковое пространство** на основе информации про **пользователей и айтемы**
- › Ищем функцию  $f(u, i)$  выдающую скор (e.g., вероятность лайка) для любой пары (user, item)
- › На основе этого делаем рекомендации



# Рекомендации - это ML и эвристики

| Не можем применять сложные модели над всем каталогом.

Делаем **многостадийные** рекомендации:

- > **Отбор кандидатов.** Набираем кандидатов более простыми моделями и эвристиками
- > **Ранжирование.** Рангируем отобранных кандидатов сложными моделями
- > На практике стадий ещё больше

**Рекомендации**



# Терминология

- › **Айтем** – объект, который рекомендуем. Элемент (англ. item) каталога
- › **Эмбеддинг** – почти синоним слова вектор. “Вкладываем” (англ. to embed) объекты в векторное пространство. Точки в этом пространстве – эмбеддинги
- › **Импрешн** – объект, которые мы реально показали пользователю. Он оставил какой-то отпечаток, след, впечатление (англ. impression)
- › **Обучаемый эмбеддинг** – когда в параметрах модели “вшиты” эмбеддинги для каких-то сущностей; альтернативный подход – отдельный энкодер

# Генерация кандидатов

Примеры простых алгоритмов:

- › **item2item**: похожие на просмотренные айтемы
- › **user2item**: похожие пользователи → понравившиеся им айтемы
- › **топ-популярного**: самые популярные айтемы
- › **история пользователя**

# Ранжирование

Цель: упорядочить кандидатов по релевантности

- › Вход: 100–1000 объектов из кандидата
- › Выход: score (e.g., вероятность лайка)

Примеры моделей:

- › Градиентные бустинги
- › Нейросети

В качестве признаков берем:

- › User-features (возраст, регион, активность...)
- › Item-features (тематика, новизна...)
- › User×Item-features (история взаимодействий, similarity...)

# Матрица релевантностей

Матрица взаимодействия между пользователями и айтемами  $R \in \mathbb{R}^{|U| \times |I|}$ :

$$R_{ui} = \begin{cases} 1, & \text{если пользователь } i \text{ взаимодействовал с айтемом } i \\ 0, & \text{иначе} \end{cases}$$

**Матрица  $R$  также называется матрицей релевантностей.** Эту матрицу мы составляем сами, например, из explicit feedback (рейтинги, лайки), или implicit feedback (прослушивания, клики и проч.).

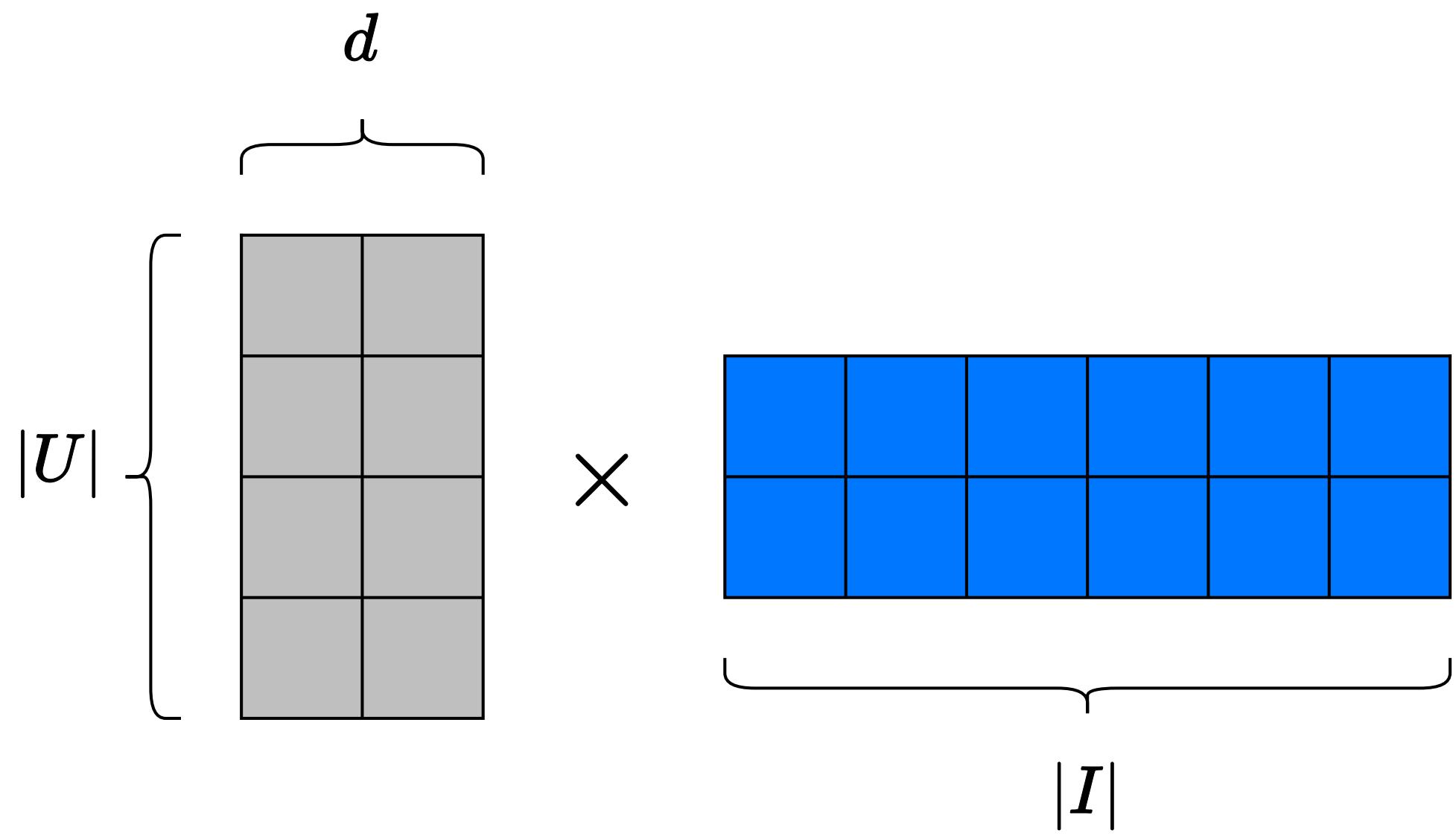
Матрица  $R$  не известна целиком. Нас интересуют пропущенные значения, которые не равны 0.

# Матричная факторизация

Чтобы восстановить часть пропущенных значений в матрице  $R$ , можно сделать матричную факторизацию:

$$R = V_U V_I^T, \text{ где } V_U \in \mathbb{R}^{|U| \times d}, V_I \in \mathbb{R}^{|I| \times d}$$

Это равносильно тому, чтобы мы учили **эмбеддинги** для каждого пользователя и айтема и приближаем значения  $R_{ui}$  с помощью скалярного произведения  $\langle v_u, v_i \rangle$ , где  $v_u$  – эмбеддинг пользователя, а  $v_i$  – эмбеддинг айтема

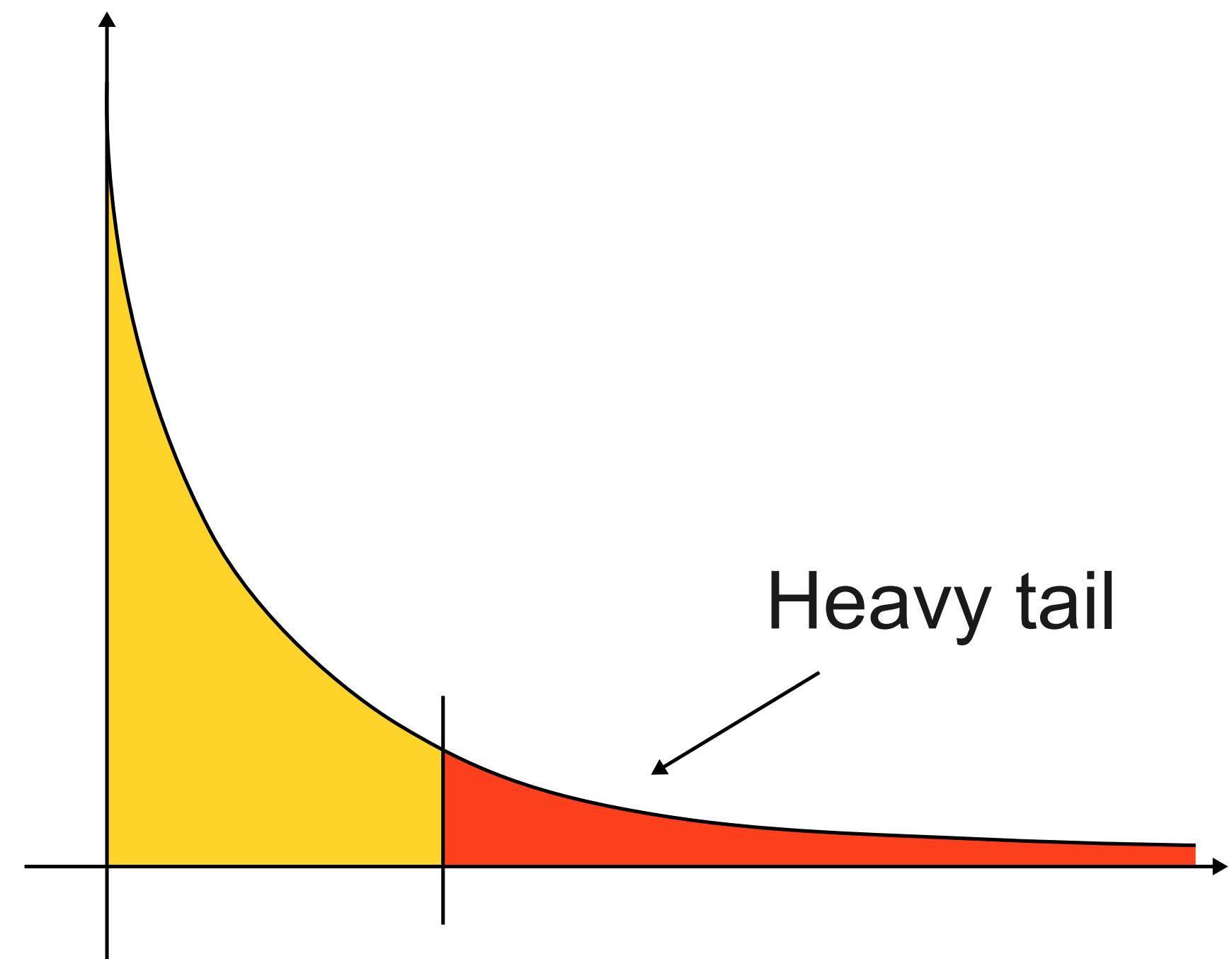


Популярный алгоритм – ALS.

NCF (Neural Collaborative Filtering) – учат MLP как “функцию близости” вместо скалярного произведения.

# Проблема тяжелого хвоста

- Нейросети учатся как запоминать наблюдаемые данные (**меморизация**), так и находить обобщенные закономерности (**генерализация**).
- Распределения объектов имеют тяжелые хвосты (**long-tailed distributions**) – большая часть значений встречается очень редко.
- Для редких объектов сложно вывести какие-то общие паттерны (генерализация), но их просто запомнить (меморизация).



2

>

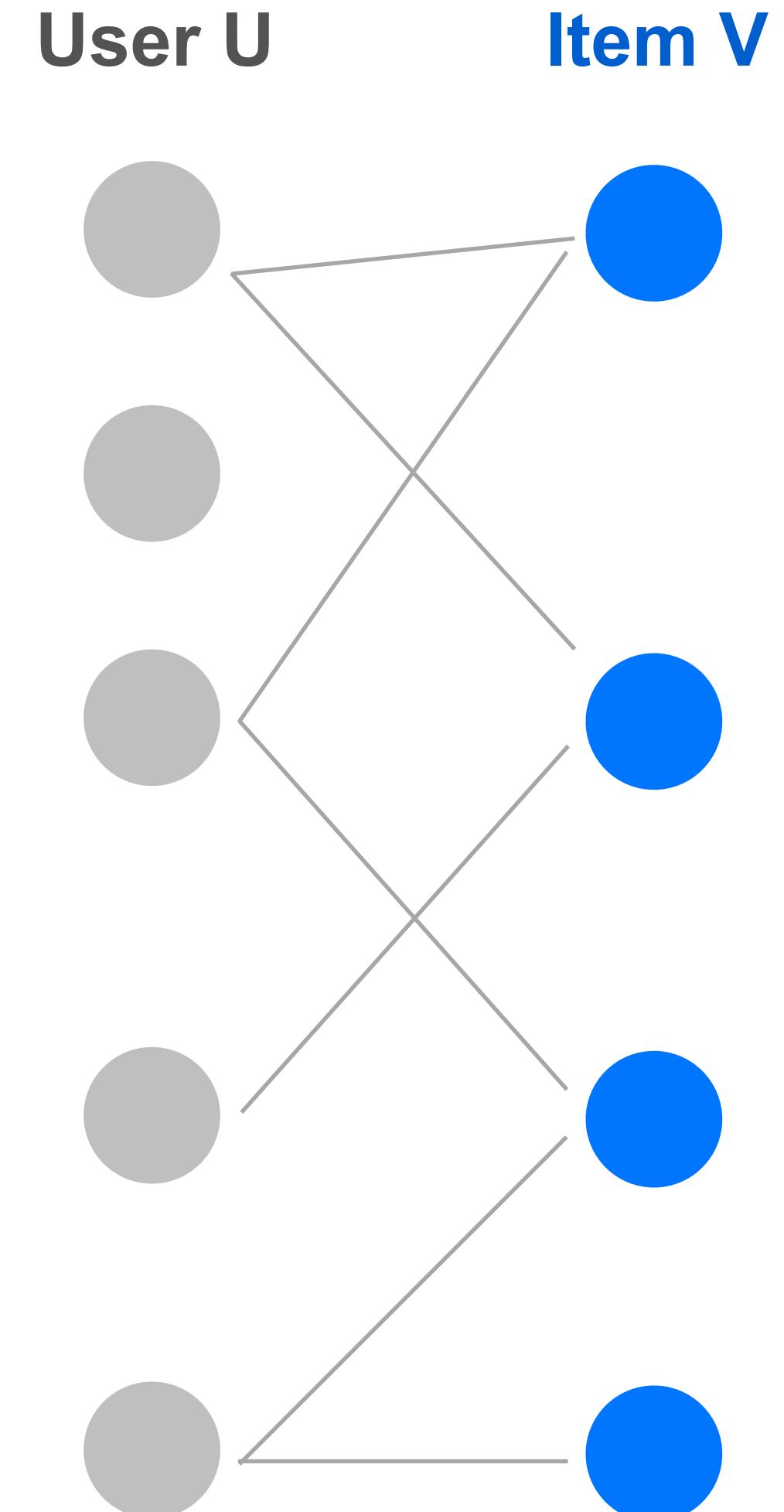
## Графовая постановка

# User-Item граф как основа рекомендаций

Мы можем рассматривать задачу рекомендаций как задачу предсказания рёбер в двудольном user-item графе.

- › Вершины  $U \cup V$ :  
 $U$  – множество всех пользователей  
 $V$  – множество айтемов
- › Рёбра  $E$  – это взаимодействия пользователей и айтемов (клик/лайк/просмотр):  
$$E = \{(u, v) \mid u \in U, v \in V, u \text{ провзаимодействовал с } v\}$$

Теперь задача сводится к **link prediction**: какие рёбра появятся в будущем?



# Матрица релевантностей = матрица смежности

Мы вводили матрицу релевантностей  $R \in \mathbb{R}^{|U| \times |I|}$ .

Каждый лайк, просмотр или другое взаимодействие – это единица в матрице и одновременно **ребро в графе**.

| То, что мы раньше назвали **матрицей релевантностей** – это и есть **матрица смежности** в user-item графе

Мы можем использовать графовые алгоритмы и GNN для извлечения векторных представлений пользователей и айтемов и предсказаний новых связей.

		Adjacency matrix $A$	
		User	Item
User	0	$R$	
Item	$R^T$		0

Из лекции [CS224W GNNs for recommender systems](#)

# Ограничения классической CF

В классических методах коллаборативный фильтрации мы обучаем эмбеддинги для каждого пользователя и айтема через приближение скалярного произведения:

$$R \approx V_U V_I^T$$

Такие модели с “неглубокими” архитектурами называют **shallow**.

- В shallow модели коллаборативный сигнал (связи в user-item графе) учитываются только неявно, через задачу моделирования.

Наглядно это видно на уровне матриц: матрица смежности  $A$  хранит только прямые связи, а в  $A^2$  появляются единицы там, где две вершины соединены двухшаговым путём.

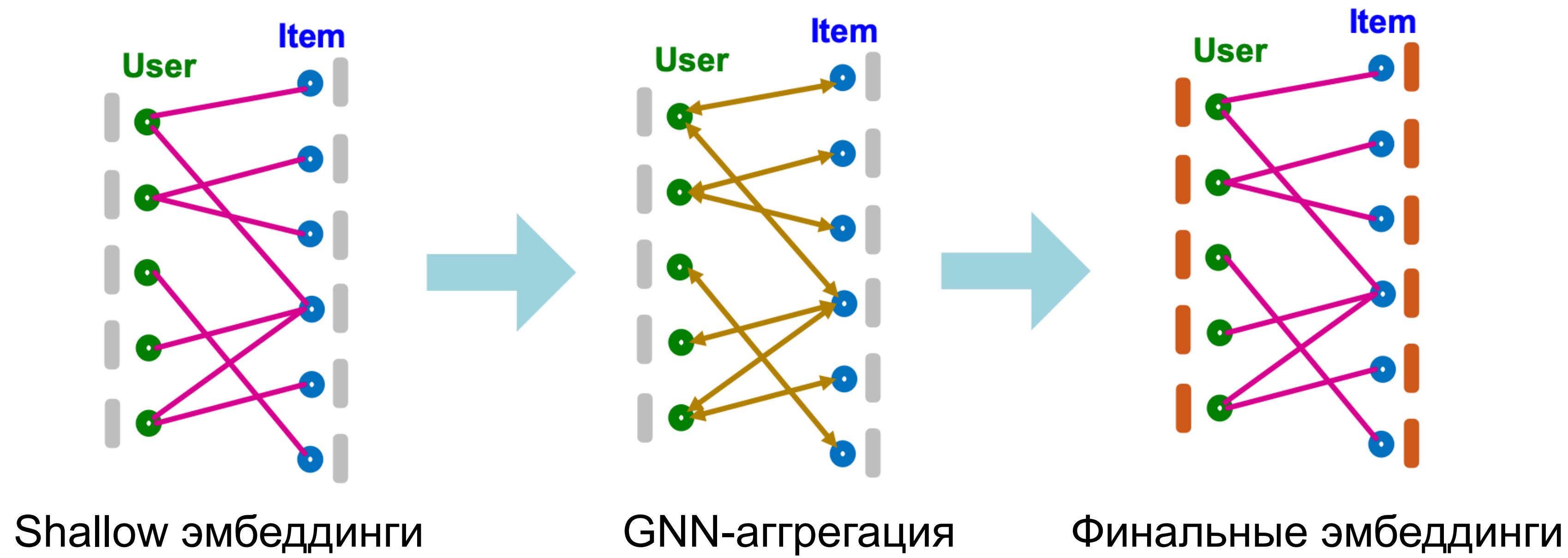
Классический MF оптимизирует только  $A$ , но игнорируя  $A^2$  и выше.

Графовые модели явно агрегируют многошаговую информацию из  $A^K$ .

А ещё — улучшаем качество на тяжелом хвосте за счет сглаживания.

# Neural Graph Collaborative Filtering

Будем явным образом интегрировать структуру двудольного user-item графа в процесс построения эмбеддингов через агрегацию соседей.



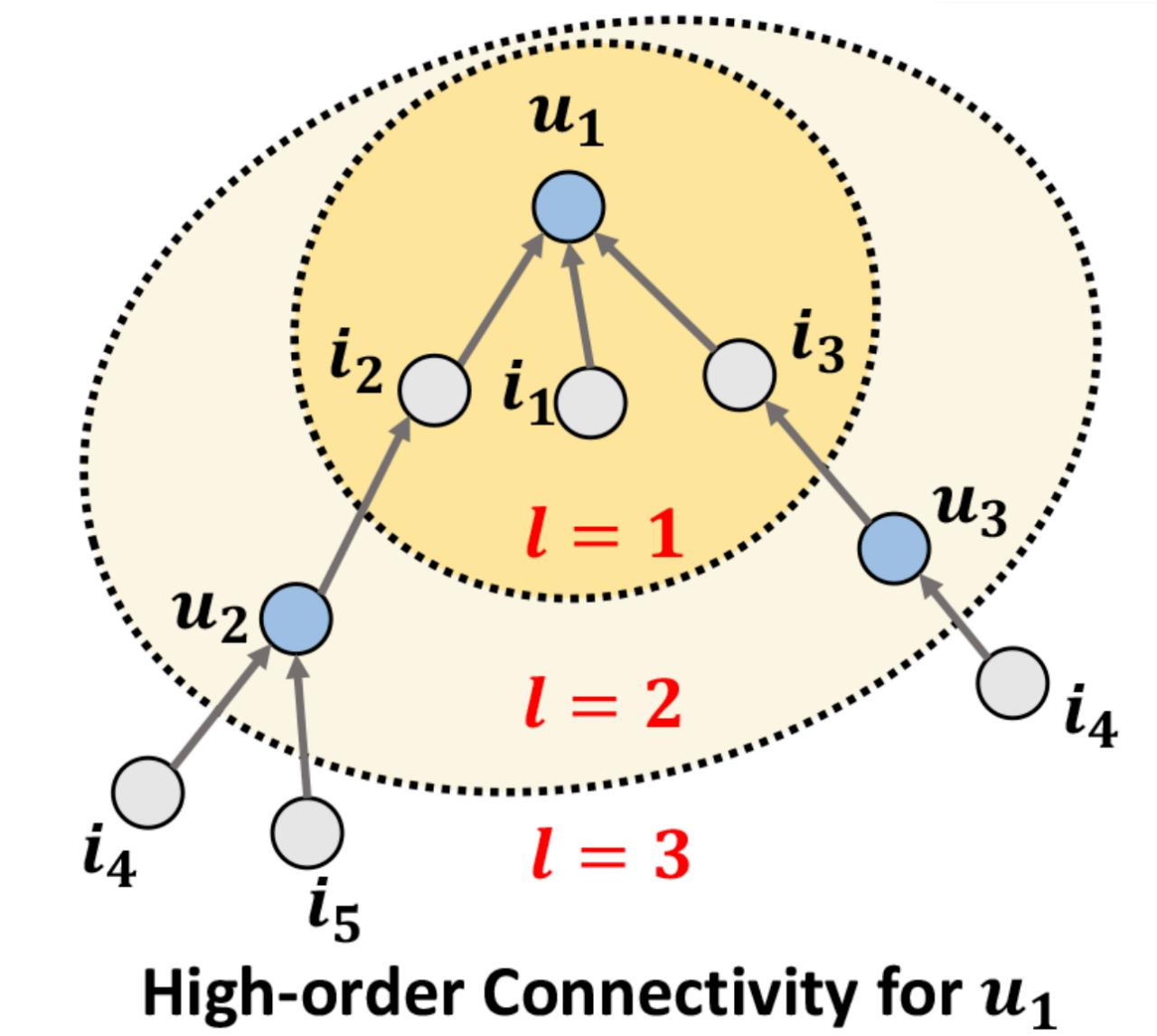
Из лекции [CS224W GNNs for recommender systems](#)

# Как работает NGCF

- Эмбеддинг вершины пересчитывается на основе ее предыдущего эмбеддинга и агрегации соседей в графе:

$$\mathbf{e}_u^{(k+1)} = \sigma \left( \mathbf{W}_1 \mathbf{e}_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u| |\mathcal{N}_i|}} \left( \mathbf{W}_1 \mathbf{e}_i^{(k)} + \mathbf{W}_2 \left( \mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)} \right) \right) \right),$$

$$\mathbf{e}_i^{(k+1)} = \sigma \left( \mathbf{W}_1 \mathbf{e}_i^{(k)} + \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u| |\mathcal{N}_i|}} \left( \mathbf{W}_1 \mathbf{e}_u^{(k)} + \mathbf{W}_2 \left( \mathbf{e}_u^{(k)} \odot \mathbf{e}_i^{(k)} \right) \right) \right),$$



- Изначальные эмбеддинги вершин (до агрегаций) — обучаемые; Также есть матрицы с весами агрегаций  $W_1, W_2$
- Чтобы сформировать финальный эмбеддинг, конкатенируем эмбеддинги со всех слоев  $\mathbf{e}^{(0)}, \mathbf{e}^{(1)}, \dots, \mathbf{e}^{(K)}$

Из статьи [Neural Graph Collaborative Filtering](#)

# Как работает NGCF

- › Чтобы оценить скор ребра используется скалярное произведение:

$$\hat{y}_{NGCF}(u, i) = e_u^{*T} e_i^*$$

где  $e_u^* = \text{CONCAT}[e_u^{(0)}, \dots, e_u^{(L)}]$ ,  $e_i^* = \text{CONCAT}[e_i^{(0)}, \dots, e_i^{(L)}]$

- › Используется **BPR loss**: скор существующего ребра должен быть выше, чем скор несуществующего ребра:

$$L = -\log \sigma(\hat{y}(u, i) - \hat{y}(u, n))$$

где  $(u, i)$  - существующее взаимодействие из данных,  $(u, n)$  - случайно сэмплируется

# NGCF: результаты

**Table 2: Overall Performance Comparison.**

	Gowalla		Yelp2018*		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
MF	0.1291	0.1109	0.0433	0.0354	0.0250	0.0196
NeuMF	0.1399	0.1212	0.0451	0.0363	0.0258	0.0200
CMN	<u>0.1405</u>	<u>0.1221</u>	0.0457	0.0369	0.0267	0.0218
HOP-Rec	0.1399	0.1214	<u>0.0517</u>	<u>0.0428</u>	<u>0.0309</u>	<u>0.0232</u>
GC-MC	0.1395	0.1204	0.0462	0.0379	0.0288	0.0224
PinSage	0.1380	0.1196	0.0471	0.0393	0.0282	0.0219
<b>NGCF-3</b>	<b>0.1569*</b>	<b>0.1327*</b>	<b>0.0579*</b>	<b>0.0477*</b>	<b>0.0337*</b>	<b>0.0261*</b>
%Improv.	11.68%	8.64%	11.97%	11.29%	9.61%	12.50%
<i>p</i> -value	2.01e-7	3.03e-3	5.34e-3	4.62e-4	3.48e-5	1.26e-4

# LightGCN

- › Сильно упростили модель относительно NGCF, улучшив качество:

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|} \sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)}$$

$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|} \sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}$$

- › В качестве обучаемых параметров только обучаемые эмбеддинги пользователей и айтемов
- › Вместо конкатенации  $\mathbf{e}^{(0)}, \mathbf{e}^{(1)}, \dots, \mathbf{e}^{(K)}$  используется сумма. Нельзя оставить только  $k$ -й вектор из-за over-smoothing'а (каждый новый слой свертки увеличивает over-smoothing).
- › Учатся тоже на BPR

# LightGCN: результаты

**Table 4: The comparison of overall performance among LightGCN and competing methods.**

<b>Dataset</b>	<b>Gowalla</b>		<b>Yelp2018</b>		<b>Amazon-Book</b>	
<b>Method</b>	<b>recall</b>	<b>ndcg</b>	<b>recall</b>	<b>ndcg</b>	<b>recall</b>	<b>ndcg</b>
NGCF	0.1570	0.1327	0.0579	0.0477	0.0344	0.0263
Mult-VAE	0.1641	0.1335	0.0584	0.0450	0.0407	0.0315
GRMF	0.1477	0.1205	0.0571	0.0462	0.0354	0.0270
GRMF-norm	0.1557	0.1261	0.0561	0.0454	0.0352	0.0269
<b>LightGCN</b>	<b>0.1830</b>	<b>0.1554</b>	<b>0.0649</b>	<b>0.0530</b>	<b>0.0411</b>	<b>0.0315</b>

3

>

# PinSage

# Recap: GraphSage

**GraphSage** (SAmple and aggreGatE) — индуктивный GCN:

- › Для каждой вершины есть какие-то “сырые” признаки, которые можем закодировать,  $h_v^0 \leftarrow x_v$
- › Равномерно **сэмплируем** соседей и **агрегируем** их эмбеддинги:

$$h_{N(v)}^k \leftarrow \text{AGGREGATE}_k \left( h_y^{k-1}, \forall u \in N(v) \right)$$
$$h_v^k \leftarrow \sigma \left( W^k \cdot \text{CONCAT}(h_v^{k-1}, h_{N(v)}^k) \right)$$

где  $N(v)$  - сэмплированные соседи.

- › Лосс учитывает графовую структуру: приближаем эмбеддинги близких вершин:

$$L(z_u) = -\log(\sigma(z_u^T z_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(z_u^T z_{v_n}))$$

где  $v$  — вершина, который встречается вблизи  $u$  при случайному блуждании фиксированной длины

- › Применили не для рекомендаций, а для node-level classification

# Pinterest

Созданные



Все пины

33,5 тыс. пина · 4 нед.



HOME

4 тыс. пина · 4 нед.

Сохраненные



ILLUSTRATION

2 тыс. пинов · 1 мес.



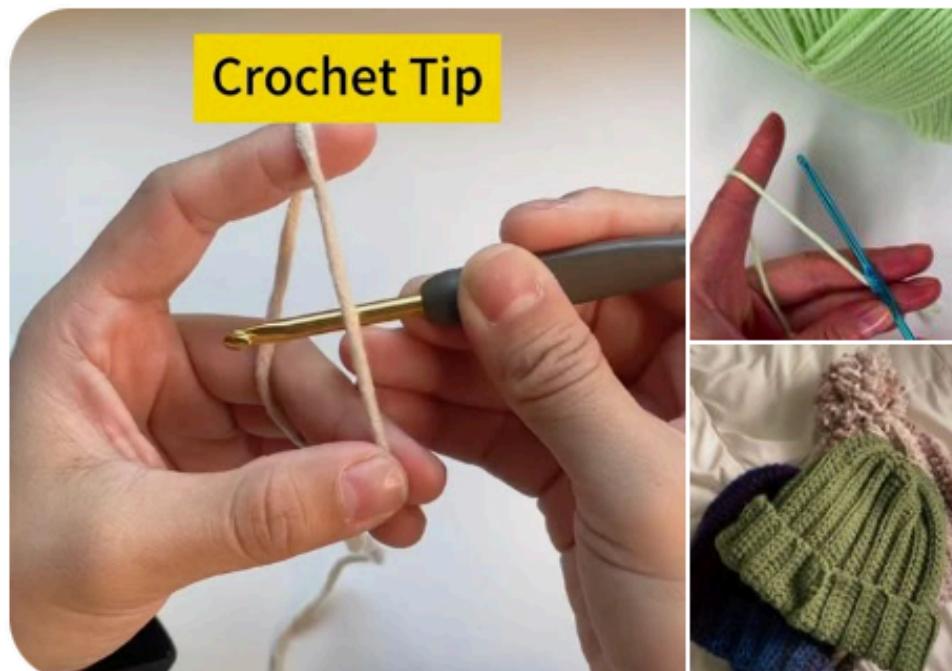
STYLE

7,3 тыс. пин · 2 мес.



CERAMIC

465 пинов · 5 мес.



crochet

3 пина · 5 мес.



NAILS

294 пина · 7 мес.



CATS & CO

275 пинов · 7 мес.



# Pinterest

**Pinterest** - это интернет-сервис, предназначенный для поиска, сохранения и обмена визуальным контентом (изображениями и видео).

Пользователи создают тематические подборки (**доски/boards**), куда они сохраняют понравившиеся изображения (**пины**).

Уже на момент 2018 года:

- › Более 2 млрд пинов
- › Более 1 млрд досок
- › Более 18 млрд ребер (сохранений пинов на различные доски)

**Pinterest - рекомендательная платформа:**

- › Главная страница - это рекомендательная лента пинов (pin to user)
- › К пинам рекомендуются похожие пины (pin to pin)
- › К доскам рекомендуются подходящие пины (pin to board)

# Pinterest

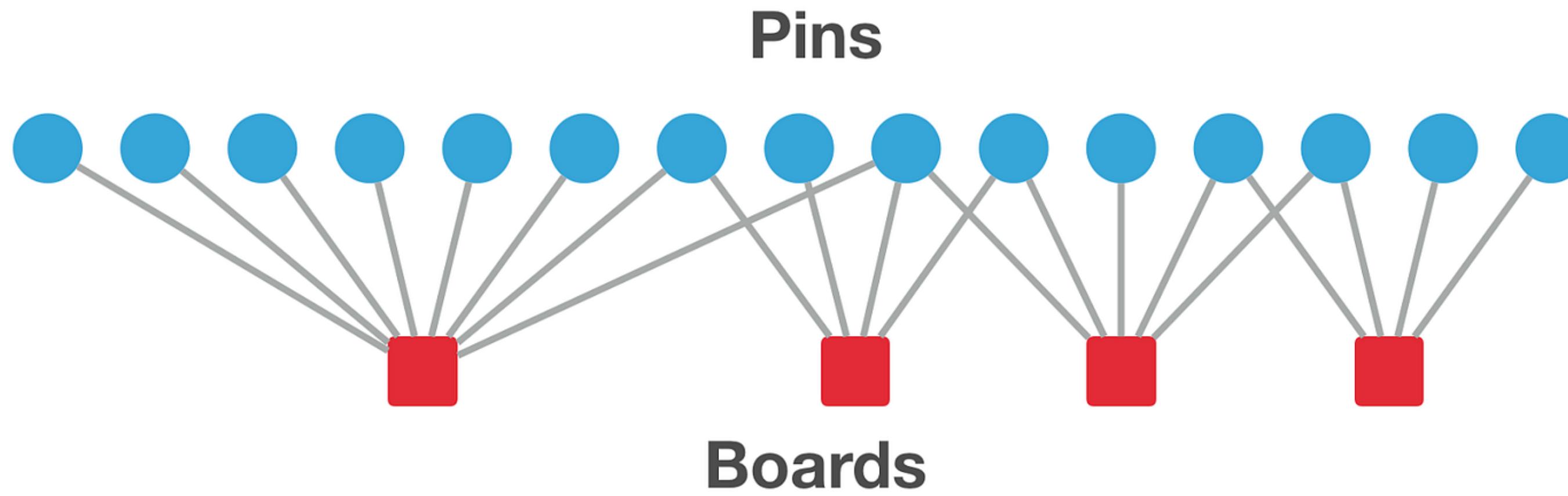
На момент разработки PinSage:

- › **PinText**: векторные представления пинов на основе названия и описания с помощью word2vec
- › Есть визуальный эмбеддинг пина с помощью VGG, обученного на классификацию
- › **Pixie<sup>1</sup>**: случайные блуждания на pin-board графе, используемые для рекомендаций
- › Позже появятся **нейросетевое ранжирование и двухбашенные нейросети для генерации кандидатов**

<sup>1</sup> [Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time](#)

# Pin-board граф

Очень большой двудольный pin-board граф с вершинами двух типов:  $I$  (пины) и  $C$  (доски):



# Pixie

- › Начинаем блуждания из query pin'a
- › Во время блужданий считаем node visit counts
- › Отбираем в качестве рекомендаций top по visit counts
- › **Biased random walk:** при блужданиях выбираем ребра не случайно, а исходя из релевантности пользователю (смотрим на язык, тему пина) присваиваем веса и сэмплируем пропорционально ним

# Pixie

- › Чем глубже блуждания, тем разнообразней кандидаты
- › Время блужданий не зависит от размера графа
- › Для вершин с большой степенью нужно блуждать больше
- › Чтобы сделать рекомендации для пользователя, берут несколько его пинов и для каждого запускают блуждания
- › Внедрили в рекомендацию похожих пинов и на homefeed

# PinSage

- › Применили **GraphSAGE** к pin-board графу: все авторы GraphSage являются также авторами PinSage
- › **Importance-based sampling**: поменяли алгоритм сэмплирования соседей. В качестве соседей используют топ посещенных вершин из случайных блужданий с помощью Pixie
- › **Importance pooling**: Нормализованные visit counts используются при агрегации векторов соседей
- › Вместо манипуляций полной матрицей смежности при обучении, формируют подграф для вершин, попавших в батч и для него на лету всё считают

# PinSage

- › В отличие от GraphSAGE, учатся supervised на отдельно отобранные пары пинов (подряд идущие пины с хорошим откликом от пользователя):

$$L(q, p) = \max\{0, \langle q, p \rangle - \langle q, n \rangle + \Delta\}$$

- › **Curriculum learning:** негативы сначала сэмплируются равномерно из всего каталога, затем с помощью PPR (personalized page rank)

Внедрили и в related pins, и на homefeed. Индуктивность: можем сходу считать векторы для новых пинов, на которых не обучались.

# PinSage: результаты

Query



Visual only



PinSAGE



# PinSage: результаты

Methods	Win	Lose	Draw	Fraction of wins
PinSage vs. Visual	28.4%	21.9%	49.7%	56.5%
PinSage vs. Annot.	36.9%	14.0%	49.1%	72.5%
PinSage vs. Combined	22.6%	15.1%	57.5%	60.0%
PinSage vs. Pixie	32.5%	19.6%	46.4%	62.4%

**Table 2: Head-to-head comparison of which image is more relevant to the recommended query image.**

4



# Продвинутые рекомендательные системы

# Рекомендации — это DL

Нейросети идеально подходят для рекомендательных систем:

- › Много неструктурированных данных: истории пользователей тексты, изображения, истории пользователей
- › Огромные объёмы поведенческих данных — триллионы user-item взаимодействий
- › Важно обрабатывать контент, чтобы эффективно работать с **новыми и редкими** айтемами, а также показывать хорошие рекомендации **холодным** пользователям

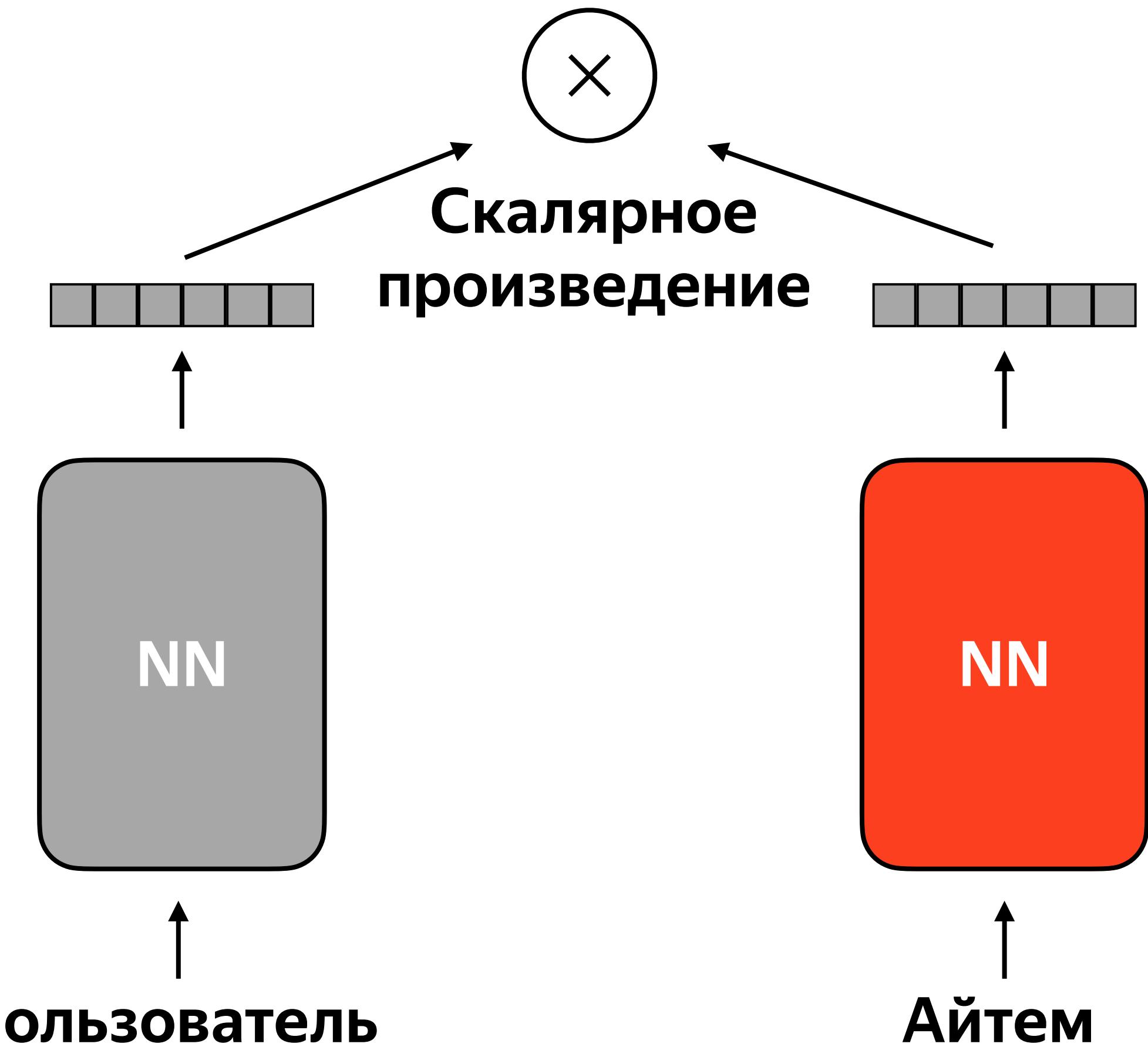
... и проникают во все стадии рекомендаций — от отбора кандидатов до ранжирования.

# Нейросетевой отбор кандидатов

Используем двухбашенные нейросети для фильтрации больших каталогов.

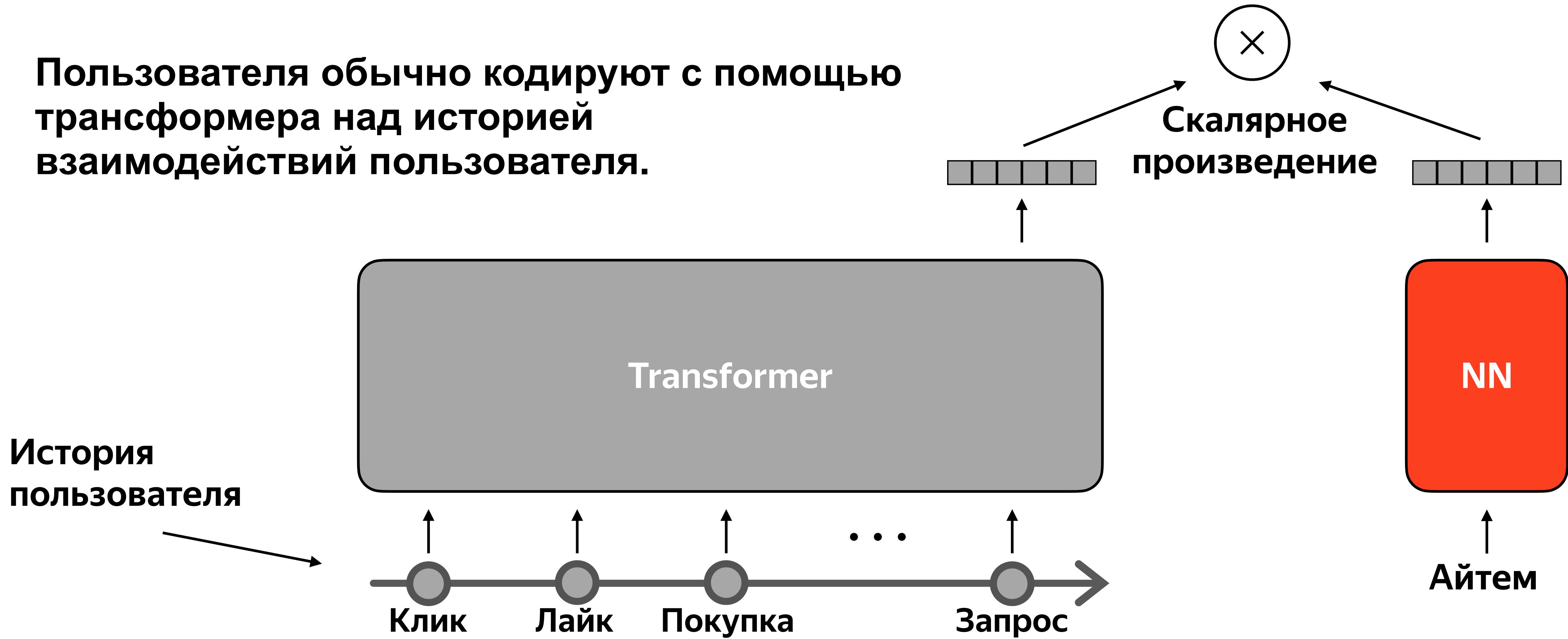
Строим семантическое пространство:

- > Эмбеддинги пользователей и айтемов из нейросетевых башен
- > Скалярное произведение — функция похожести
- > Умеем быстро искать соседей в векторном пространстве



# Нейросетевой отбор кандидатов

Пользователя обычно кодируют с помощью трансформера над историей взаимодействий пользователя.



# Обучение генератора кандидатов

Для обучения генераторов кандидатов как правило используют софтмакс по айтемам:

$$L_{\text{softmax}}(u, p) = - \log \frac{e^{f_\theta(u, p)}}{\sum_{d' \in D} e^{f_\theta(u, d')}}$$

Но не по всему каталогу, а по сэмплированным айтемам – **sampled softmax loss**:

$$L_{\text{sampled}}(u, p) = - \log \frac{e^{f_\theta(u, p)}}{e^{f_\theta(u, p)} + \sum_{i=1}^n e^{f_\theta(u, d_i)}}, \quad d_i \sim Q(d),$$

где  $u \in U, p \in D$  – позитив,  $\{d_i\}_{i=1}^n$  – сэмплированные негативы.

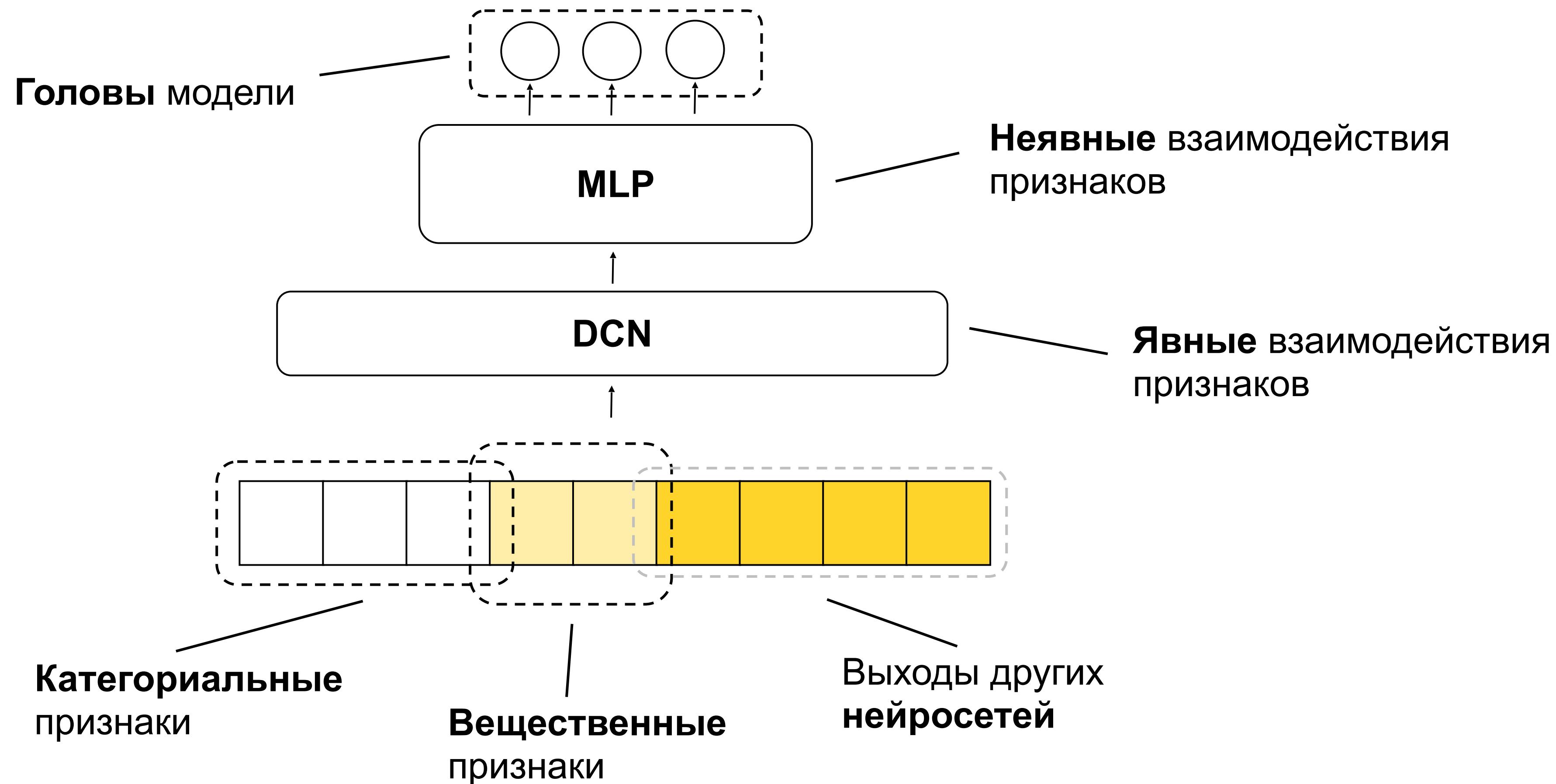
# Обучение генератора кандидатов

Часто при обучении негативы сэмплируются из батча , и чтобы модель не начинала штрафовать айтемы за популярность, используется **logQ-коррекцию**:

$$L_{\text{logQ}}(u, p) = - \log \frac{e^{f_\theta(u, p)}}{e^{f_\theta(u, p)} + \sum_{i=1}^n e^{f_\theta(u, d_i) - \log Q(d_i)}}$$

$Q(d)$  – частота айтема в данных (популярность).

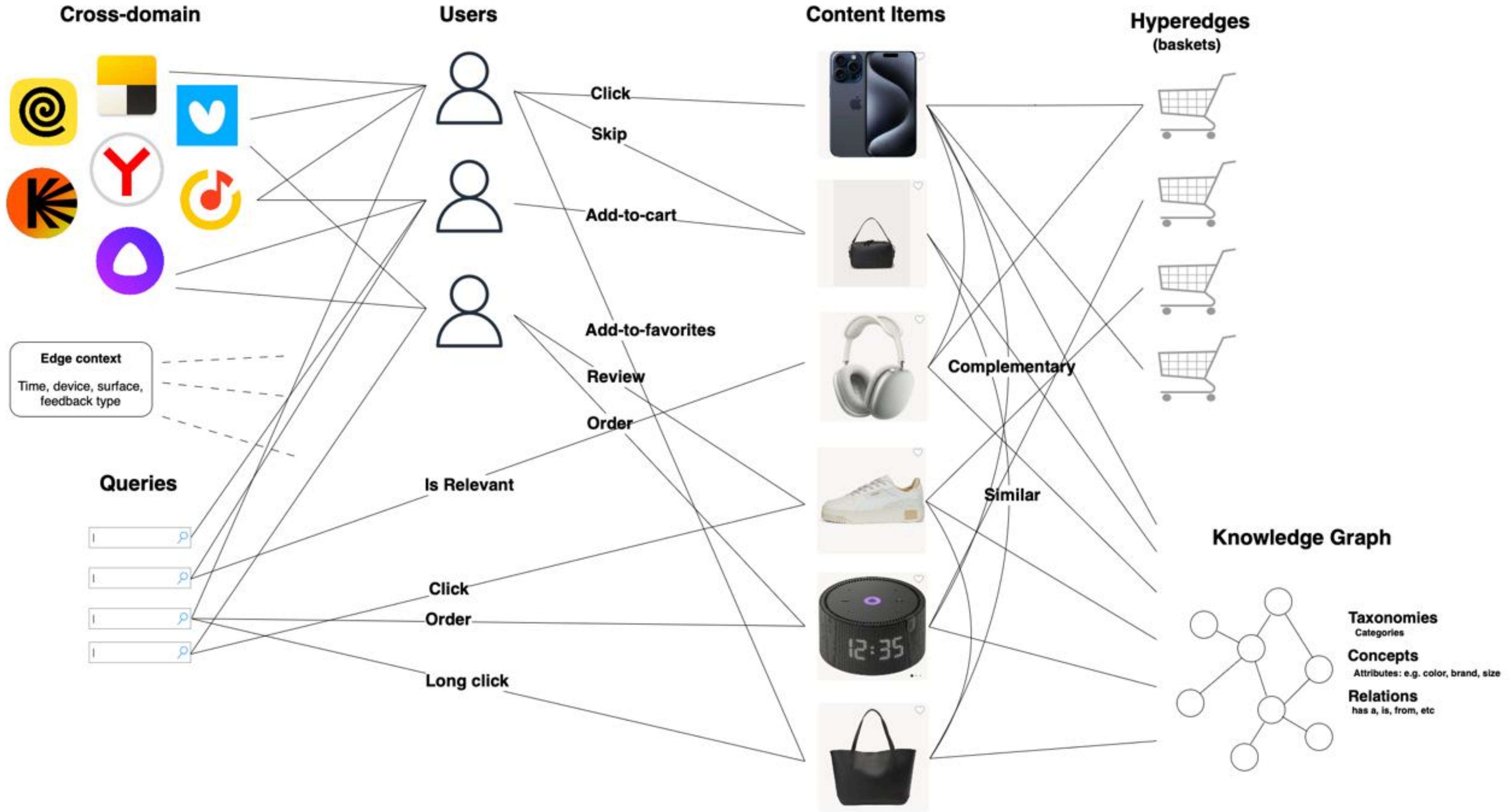
# Нейросетевое ранжирование



5

>

## **Гетерогенность, TwHIN, MultiBiSage, OmniSage**



# Гетерогенный граф

- › Двудольный user-item граф — это всего лишь часть доступной информации
- › Экосистема (или отдельный сервис) порождает большой граф, внутри которого есть разные сущности (пользователи, айтемы, запросы, реклама, концепты, кросс-домен) и отношения (разные типы взаимодействий, разные контексты взаимодействий)

**Гетерогенный граф** — есть разные типы вершин / ребер.

Хотим использовать этот граф, чтобы получить максимально хорошие, общие, фундаментальные векторные представления для всех важных нам сущностей. Сделать **representation learning** на графе!

**Link prediction** на таком гетерогенном графе - максимально общая постановка задачи для обучения.

# Графы знаний

Граф знаний — граф, вершины в котором это какие-то концепты, свойства, объекты; а ребра описывают отношения между ними, e.g. “Body Paint” -> {песня, которая была написана исполнителем} -> “Arctic Monkeys”.

- › Google Knowledge Graph: большой граф с различными фактами и информацией, который иногда помогает отвечать на вопросы в поиске
- › Также у Google есть графы знаний для Ютуба и для Google Shopping
- › Графы знаний есть у крупных екомов (Amazon, Alibaba), у Нетфликса

# TransE

**Translating Embeddings for Multi-relational Data** (TransE) — учим эмбеддинги на графах, в которых есть разные типы ребёр (отношения), e.g. на графах знаний:

- › Вершины - сущности
- › Ребра - это тройки  $(h, l, t)$ , где  $h$  - сущность head,  $t$  - сущность tail,  $l$  - отношение между head и tail сущностями

**Ключевая идея:**

$$\mathbf{h} + \mathbf{l} \approx \mathbf{t}, \text{ где}$$

$\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$  — эмбеддинги сущностей

$\mathbf{l} \in \mathbb{R}^d$  — эмбеддинг отношения

Учим на margin-based loss (как в PinSage):  $L(h, l, t, n) = \max\{0, \langle h + l, t \rangle - \langle h + l, n \rangle + \Delta\}$

# Twitter и HIN

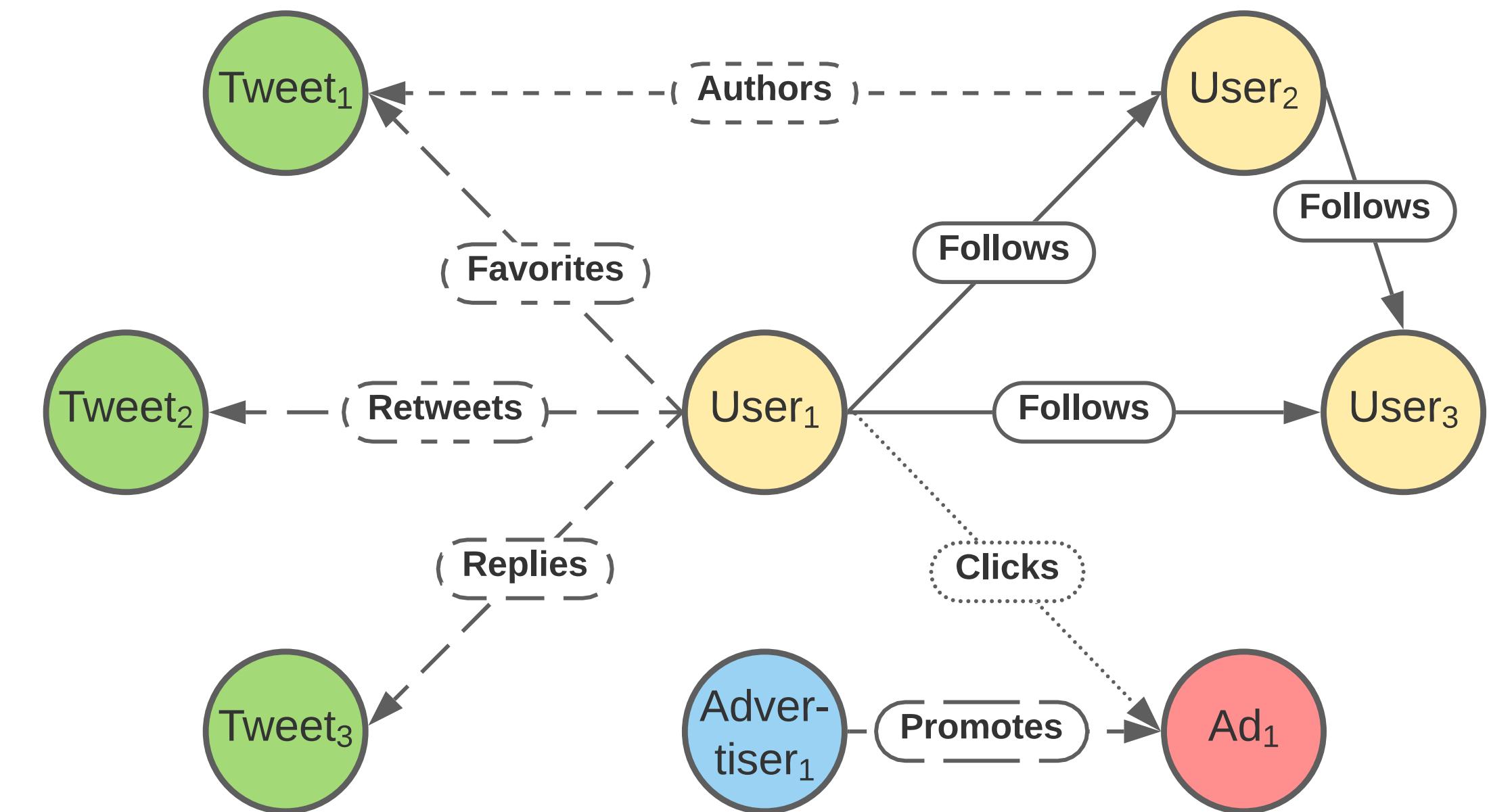
Разберем конкретный пример — как обучение на гетерогенных графах применяется в Twitter.

Социальные сети, такие как Twitter, образуют **гетерогенную информационную сеть (HIN)**:

- › Вершины: многочисленные сущности платформы (пользователи, твиты, рекламодатели и т.д.)
- › Рёбра: лайки, ретвиты, подписки, клики по рекламе, поиск

Платформа активно использует рекомендации:

- › Лента твитов
- › Показы рекламы
- › Рекомендации Who-to-Follow
- › Персонализированный поиск



Из статьи [TwHIN](#)

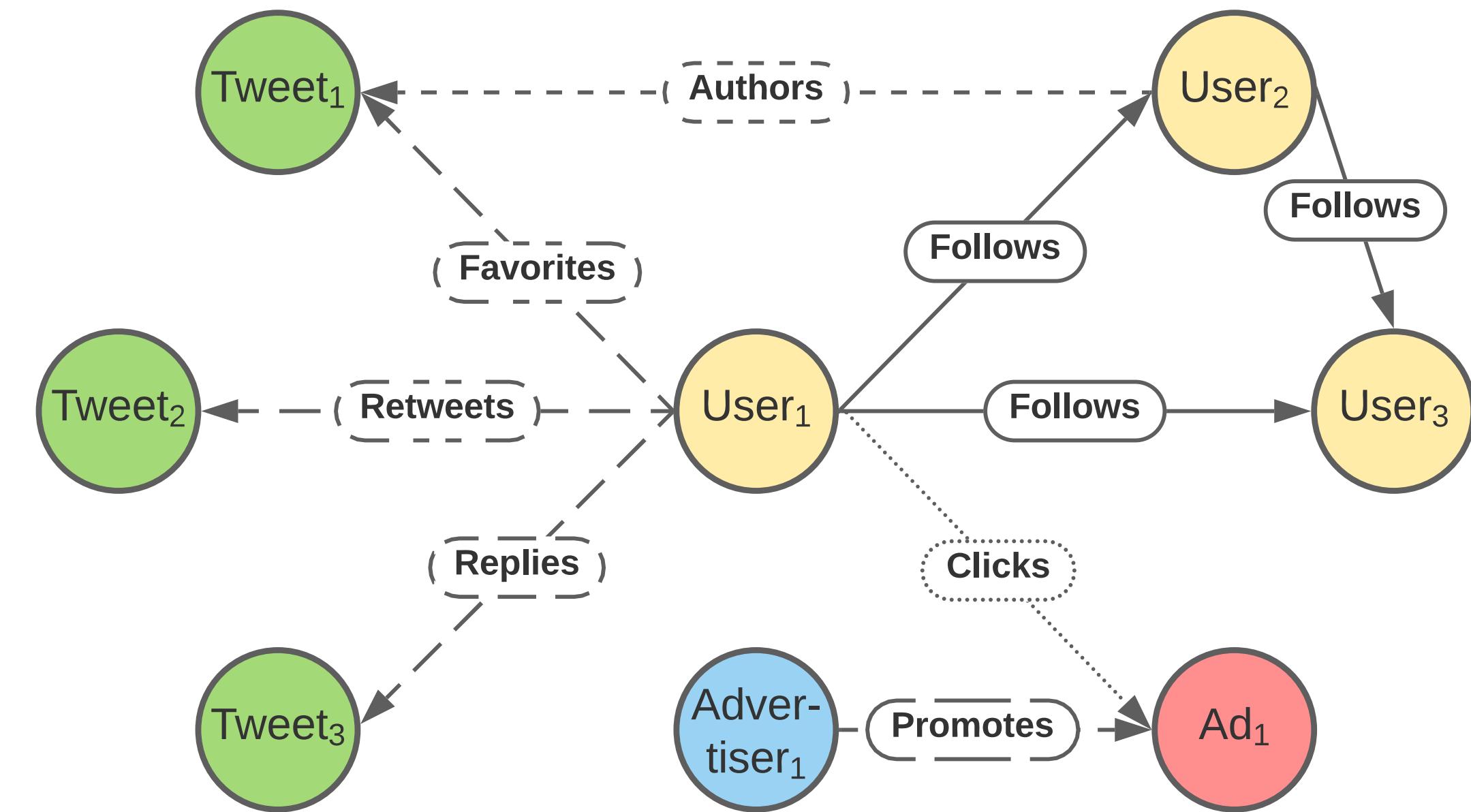
# Twitter: преимущества гетерогенного графа

В теории, можно составить разные однородные графы:

- › граф подписок (user follow graph)
- › engagement граф (user-tweet graph)

Но у гетерогенного графа есть преимущества:

- › **Больше информации**
  - разные типы взаимодействий дают дополнительные сигналы
- › **Дополнение редких событий**
  - кликов по рекламе мало, но лайков и просмотров много
- › **Универсальные эмбеддинги**
  - одна модель для ленты, поиска, рекомендации фолловеров
- › **Наиболее общая формулировка задачи**
  - link prediction между любыми сущностями



Из статьи [TwHIN](#)

# TransE для гетерогенного графа Twitter

Для ребра типа  $r$  от сущности  $s$  (source) к сущности  $t$  (target), скор этого ребра:

$$f(e) = f(s, r, t) = (\theta_s + \theta_r)^T \theta_t$$

- ›  $\theta_s, \theta_t$  – эмбеддинги сущностей
- ›  $\theta_r$  – эмбеддинг ребра типа  $r$

Задача обучения – link prediction:

$$\arg \max_{\theta} \sum_{e \in G} \left[ \log \sigma(f(e)) + \sum_{e' \in N(e)} \log \sigma(-f(e')) \right]$$

- ›  $N(s, r, t) = \{(s, r, t') : t' \in V\} \cup \{(s', r, t) : s' \in V\}$  – несуществующие ребра как негативные примеры

# TwHIN: мультимодальные эмбеддинги

- | TransE – это трансдуктивная модель: получаем эмбеддинги лишь для существующих в графе сущностей

В статье дополнительно создают мультимодальные эмбеддинги:

- > Кластеризуют унимодальные (каждого типа сущности) эмбеддинги (k-means)
- > Для каждой сущности  $u_i$  берем  $M(u_i)$  кластеров с наибольшим количеством взаимодействий
- > Считаем распределение взаимодействий с этими кластерами:

$$P(c | u_i) = \frac{\text{count}(u_i, c)}{\sum_{c' \in M(u_i)} \text{count}(u_i, c')}$$

- > Итоговый эмбед – смесь центроидов выбранных кластеров

Теперь можно генерировать эмбеддинги для новых юзеров/объектов

# TwHIN: результаты и дообучение

## Результаты:

- > Улучшили результаты в рекламе (+ написали, что если учить эмбеддинги только на данных взаимодействий с рекламой, улучшения качества нет)
- > Улучшили персонализированный поиск

Граф постоянно растет — нужно дообучать.

При переобучении с нуля новые эмбеддинги очень сильно отличаются от старых, рассматривают два решения:

- > **warm start** (инициализируем эмбеддинги результатами предыдущей модели)
- > **l2-regularization** между старыми параметрами и новыми (увеличивает количество необходимой памяти в 2 раза)

Критика PinSage и TwHIN: не было обучаемых ID-based эмбеддингов в проде на момент внедрения графовых векторов.

Model	Baseline	U	U+A	U+T	U+A+T
Ads <sub>1</sub>	21.23	21.32	21.43	21.46	<b>21.48</b>
Ads <sub>2</sub>	13.53	13.61	13.54	<b>13.63</b>	13.59
Ads <sub>3</sub>	17.11	17.26	17.27	<b>17.27</b>	17.26

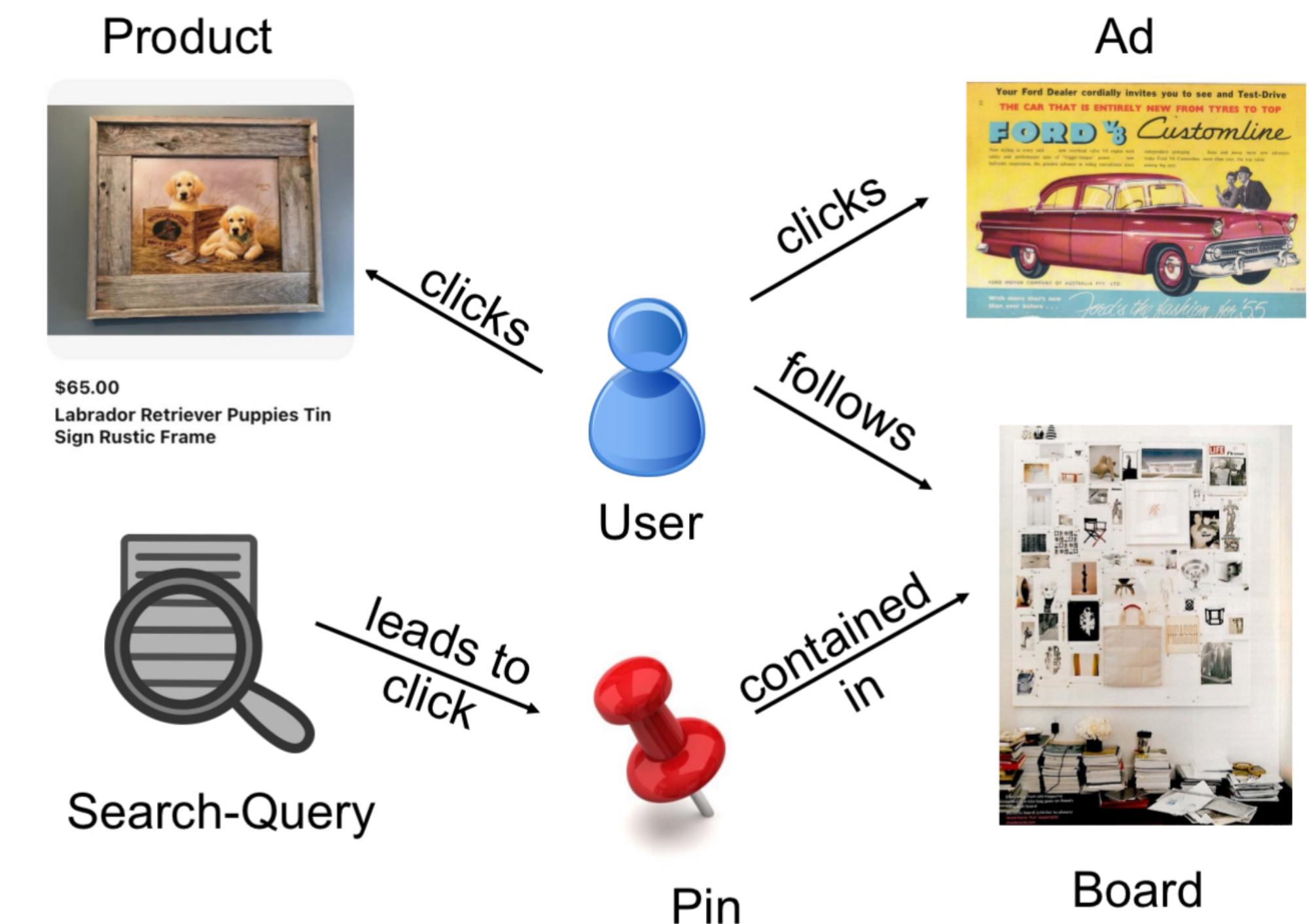
Table 2: Offline RCE for ads models with feature ablation. We investigate performance when using TwHIN embeddings for User (U), Advertiser (A), and Target entity (T) such as app to install, video to watch, or advertisement to click.

Metric	Baseline	+U <sub>f</sub>	+U <sub>e</sub>	+A	+U <sub>f</sub> + U <sub>e</sub>	+U <sub>f</sub> + U <sub>e</sub> + A
MAP	55.7	56.2	56.6	55.8	56.6	<b>57.0</b>
ROC	57.9	58.6	59.0	57.9	59.0	<b>59.6</b>

Table 3: Search engagement-based ranking with TwHIN embeddings: TwHIN user embeddings, both follow-base (U<sub>f</sub>) and engagement-base (U<sub>e</sub>), and TwHIN author embeddings (A)

# MultiBiSage

- › Взяли 6 двудольных графов: кроме pin-board добавили рекламные, поисковые данные и т.д.
- › По каждому графу отдельно собирают соседства вершины; по 50 соседей
- › Делают это не “на лету” как раньше, а перед обучением



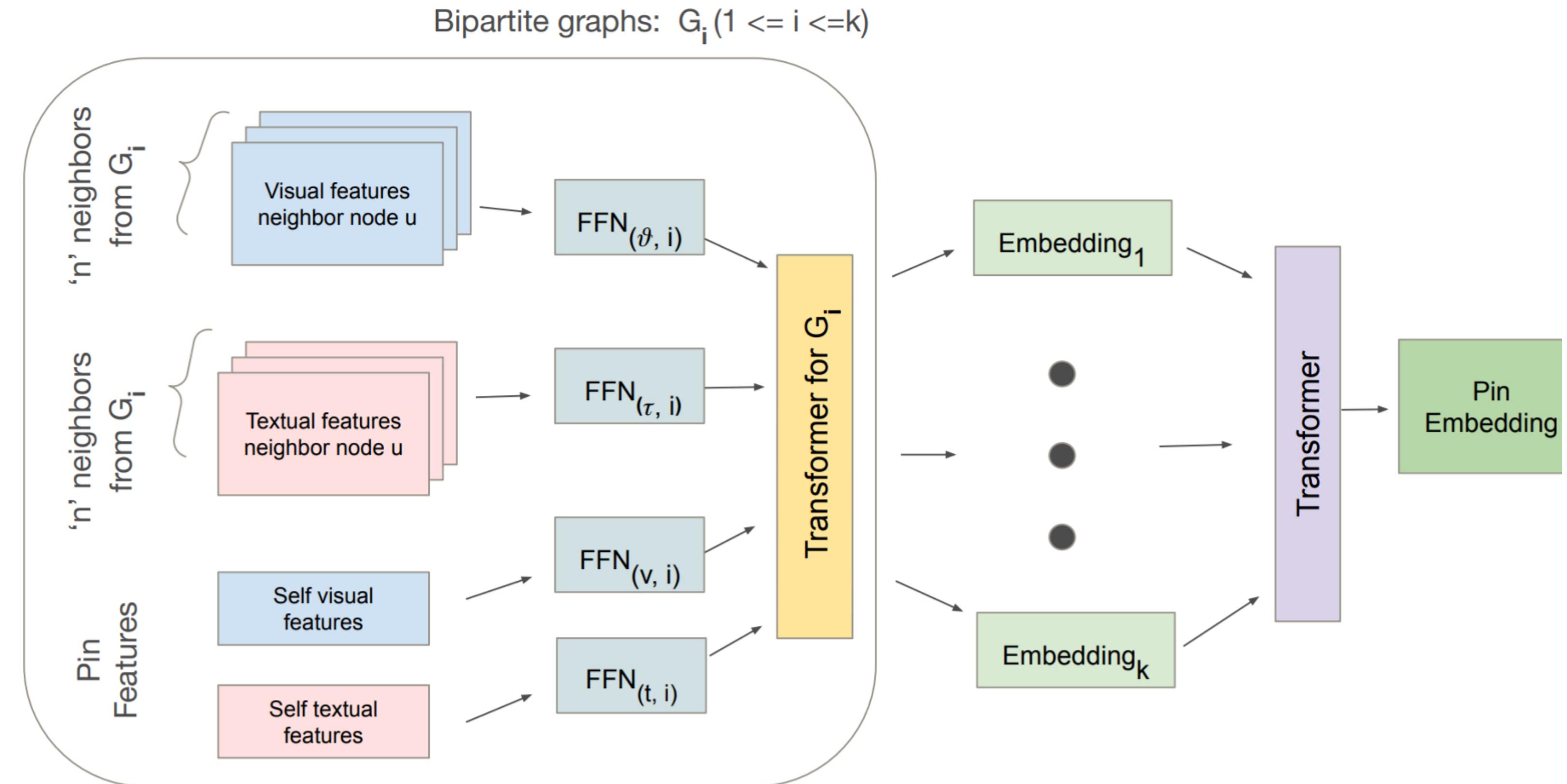
Из статьи [MultiBiSage](#)

Candidate	Graph	Rank-1	Rank-2	Rank-3	Rank-4	Rank-5
	Pin-Board					
	User-Product					
	User-Ad					
	SearchQuery-Pin					

Table 1: Candidate pin and its neighbors from diverse bipartite graphs.

# MultiBiSage

- › Прогоняют соседства из отдельных двудольных графов через свои трансформеры (вместе с целевой вершиной)
- › Получившиеся 6 эмбеддингов вершины прогоняют через еще один, “глобальный” трансформер



Из статьи [MultiBiSage](#)

# MultiBiSage

## Обучение:

- › Положительные пары набираются из item2item рекомендаций (когда пользователю понравился какой-то пин из рекомендации похожих пинов)
- › Sampled softmax loss с logq-коррекцией и mixed negative sampling:

$$L_{\text{LogQ}}(u, p) = - \log \frac{e^{f_\theta(u, p)}}{e^{f_\theta(u, p)} + \sum_{i=1}^n e^{f_\theta(u, d_i) - \log Q(d_i)}}$$

# MultiBiSage

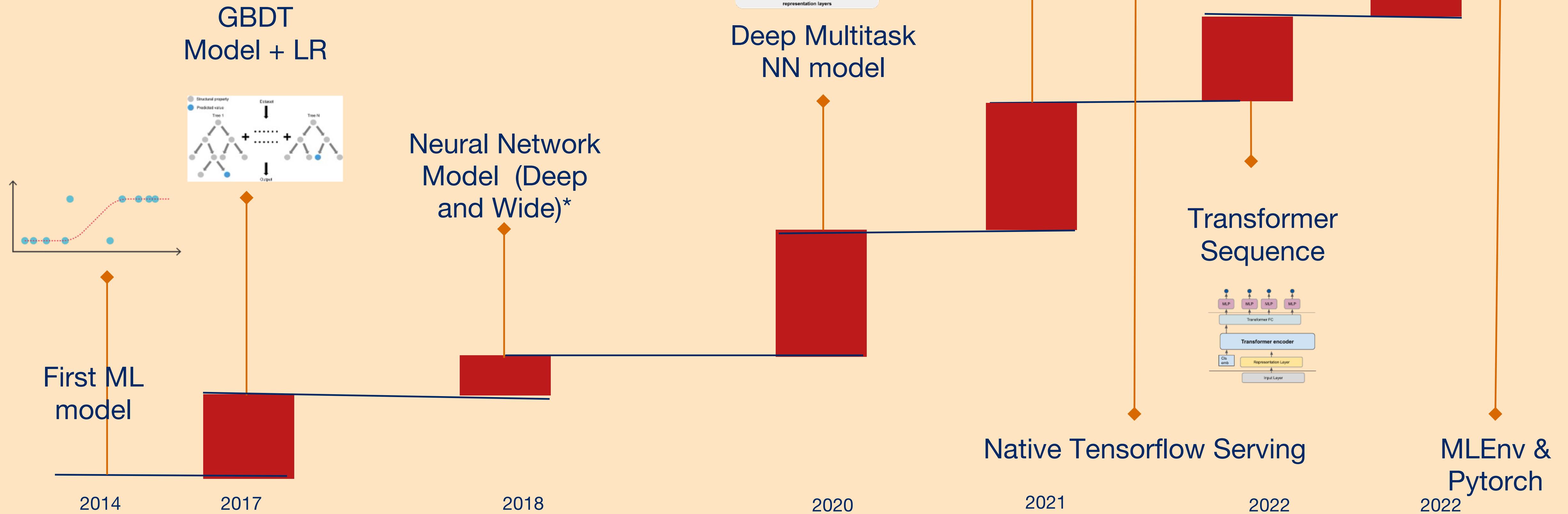
Query-Pin	Model	Engaged-Pin	Rank-1	Rank-2	Rank-3	Rank-4	Rank-5
	PinSage						
		Rank: 364					
	MultiBiSage						
		Rank: 5					

# MultiBiSage

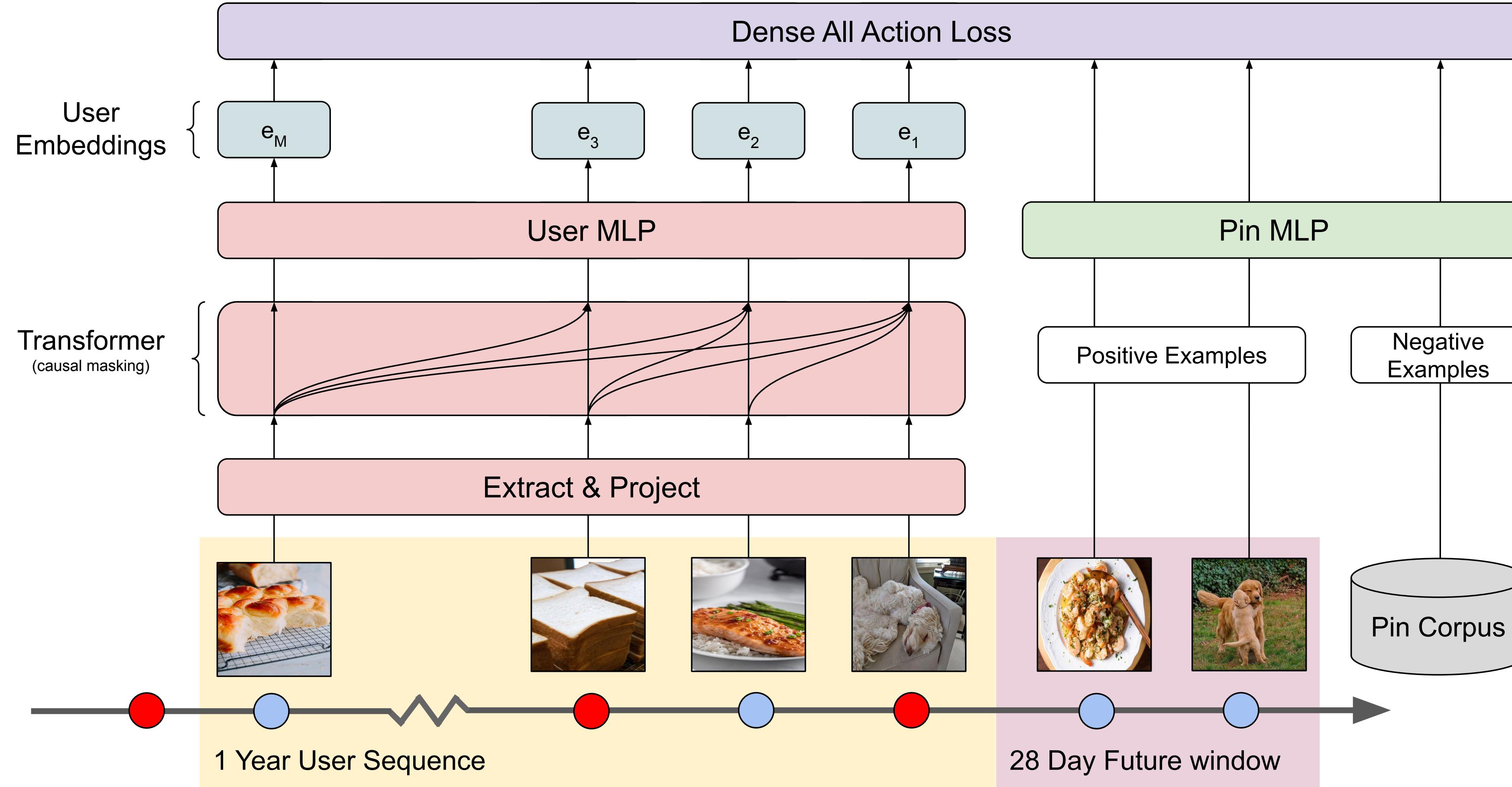
Surface/Entity Type	Engagement Type	PinSage	MultiBiSage
Organic Surface	Add-to-cart	0.893	0.907 (+1.57%)
	Checkout	0.893	0.907 (+1.57%)
Ads	Good-click-through	0.651	0.684 (+5.07%)
Related Products	Long-click	0.733	0.749 (+2.18%)
Idea Pin	Close-ups	0.724	0.776 (+7.18%)
	Repin	0.797	0.849 (+6.52%)
Video Pin	Close-ups	0.446	0.47 (+5.38%)
	Repin	0.571	0.603 (+5.60%)

**Table 4: Recommendation performance: Recall@10**

# Нейросетевое ранжирование Pinterest



# PinnerFormer



# PinnerFormer

- › Обрабатываем последние 256 событий в истории пользователя: положительные события (repin, closeup, long click) и негативные события (hide or short click)
- › Чтобы закодировать пины используем вектор из PinSage
- › Предсказываем будущие положительные взаимодействия с пинами на горизонте нескольких недель
- › Трансформер над историей пользователя — это графовая свертка

Два разных сценария внедрения:

- › Как генератор кандидатов
- › Как вектор пользователя на входе в нейросетевое ранжирование

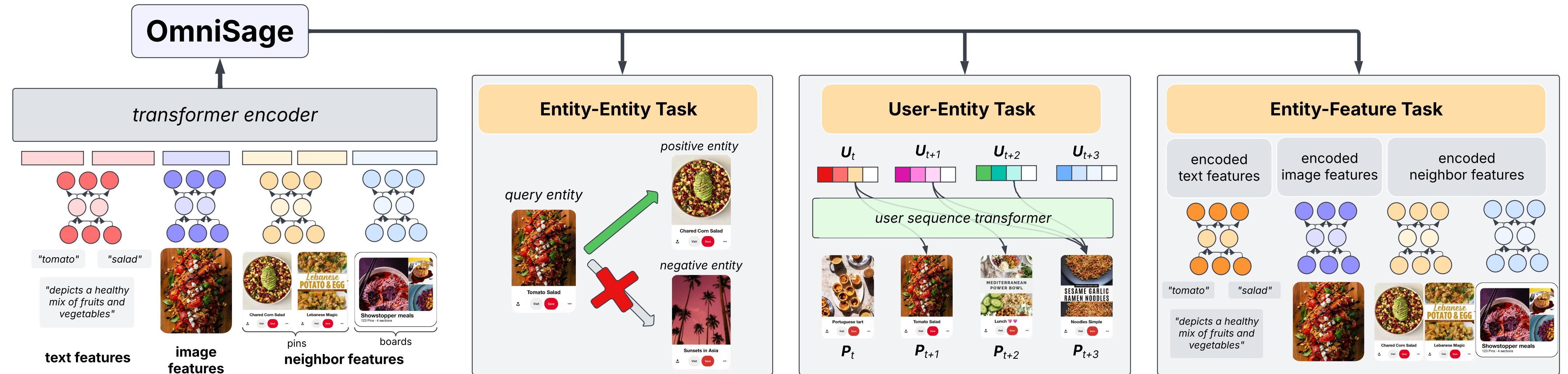
**Table 7: Online A/B experiment results replacing Pinner-Sage with PINNERTFORMER as a feature in our Homefeed ranking model. We see improvements in sitewide metrics.**

Metric	Lift	Metric	Lift
Time Spent	+1%	Homefeed Repins	+7.5%
DAU	+0.4%	Homefeed Clickthroughs	+1%
WAU	+0.12%	Homefeed Close-ups	+6%

**Table 8: Online A/B experiment results adding PINNERTFORMER as a feature to Ads ranking models. Each surface benefits significantly from PINNERTFORMER**

Metric	Related Pins	Search	Homefeed
CTR	+7.1%	+7.3%	+10.0%
gCTR	+6.9%	+5.2%	+10.1%

# OmniSage



- › Конструируют граф только из пинов и бордов
- › Добавляют pin-to-pin ребра, если в Related Pins было положительное взаимодействие
- › Для кодирования картинок используют предобученный ViT, для текста используют мешок n-gram
- › Для кодирования пинов используются OmniSage эмбеды, которые учатся end-to-end
- › Для агрегации используется однослойный трансформер с размерностью 768 и CLS-токеном
- › UserSage такого же размера, как PinnerFormer (4 слоя размерности 512)

# Про другие компании

- › Есть практика конструировать item2item граф и на нем учить что-то типа GraphSAGE (Spotify, Snapchat)
- › LinkedIn тоже использует graphsage
- › В екомах любят графы знаний (Амазон, Алибаба, Google Shopping)
- › У Нетфликса есть граф, в котором объединили граф знаний и со-engagement данные.

[LiGNN: Graph Neural Networks at LinkedIn](#)

[Towards Graph Foundation Models for Personalization](#)

[COSMO: A Large-Scale E-commerce Common Sense Knowledge Generation and Serving System at Amazon](#)

[AliCoCo: Alibaba E-commerce Cognitive Concept Net](#)

[GiGL: Large-Scale Graph Neural Networks at Snapchat](#)

[Synergistic Signals: Exploiting Co-Engagement and Semantic Links via Graph Neural Networks](#)

[CoActionGraphRec: Sequential Multi-Interest Recommendations Using Co-Action Graphs](#)

[XWalk: Random Walk Based Candidate Retrieval for Product Search](#)

# Summary

- › Обсудили рекомендательные системы, в том числе продвинутые нейросети
- › Поговорили про коллаборативный фильтрацию и графовые свертки на двудольном user-item графе
- › Обсудили PinSage и всё, что с ним связано; в том числе его дальнейшую эволюцию
- › Посмотрели на гетерогенный граф, подход Twitter'a

[LiGNN: Graph Neural Networks at LinkedIn](#)

[Towards Graph Foundation Models for Personalization](#)

[COSMO: A Large-Scale E-commerce Common Sense Knowledge Generation and Serving System at Amazon](#)

[AliCoCo: Alibaba E-commerce Cognitive Concept Net](#)

[GiGL: Large-Scale Graph Neural Networks at Snapchat](#)

[Synergistic Signals: Exploiting Co-Engagement and Semantic Links via Graph Neural Networks](#)

[CoActionGraphRec: Sequential Multi-Interest Recommendations Using Co-Action Graphs](#)

[XWalk: Random Walk Based Candidate Retrieval for Product Search](#)