

# Нейросетевые рекомендательные системы

Кирилл Хрыльченко

Старший преподаватель,  
ФКН ВШЭ

Яндекс



# О себе

- 5 лет занимался нейросетевыми рекомендательными системами в индустрии (в Яндексе)
  - Руководил R&D-командой, занимавшейся нейросетевыми рекомендациями
  - Работал с двухбашенными моделями, нейросетевым ранжированием, трансформерами
- Первый автор статей на ACM RecSys'25 и ACM KDD'26
- Преподаю RecSys в ШАДе, а теперь и в ВШЭ
- Семинар и домашнее задание подготовлены коллегами из моей бывшей R&D команды



Кирилл Хрыльченко  
(лекция)



Артём Матвеев  
(семинар)



Владимир Байкалов  
(домашнее задание)

P.S: Гёте в подготовке занятия не участвовал.

# О чём сегодня поговорим

1. Зачем вообще нужны рекомендательные системы
2. Как устроена классическая рексистема (на примере YouTube до DL)
3. Зачем в неё добавлять нейросети
4. Нейросетевые архитектуры
  - двухбашенные модели
  - нейросетевое ранжирование
5. Frontier: трансформеры, масштабирование, RL, генеративные рекомендации, разговорные рексистемы, semantic IDs

# Зачем нужны рекомендательные системы?

Часть 1



# Цифровой мир = бесконечный каталог

- **Музыка:** >100 млн треков
- **Видео:** миллиарды роликов, десятки миллионов новых в день
- **Социальные сети:** сотни миллионов постов ежедневно
- **Маркетплейсы:** десятки-сотни миллионов товаров

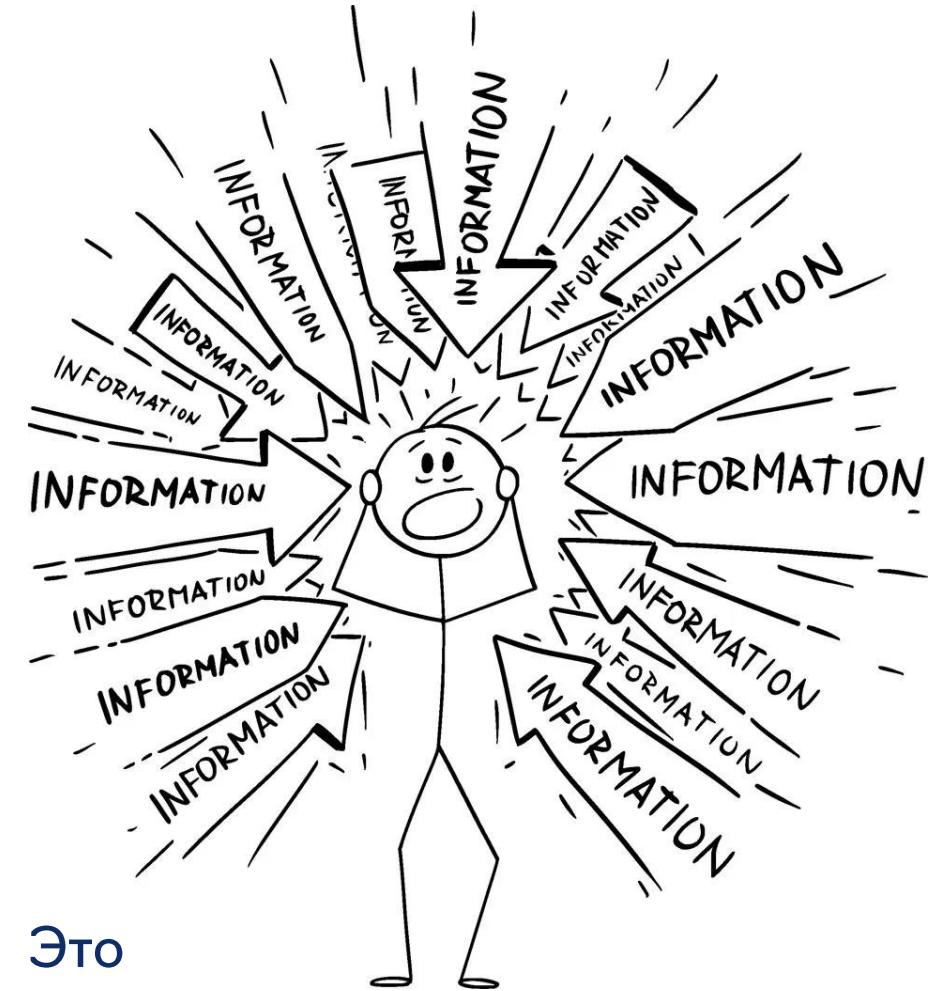


# Масштабы YouTube

2022:

- 10 млрд видео
- 500 часов нового контента каждую минуту
- 1 млрд часов просмотров в день
- 4% видео дают 94% всех просмотров

Чтобы посмотреть всё – понадобятся десятки тысяч лет.



Это

информационная  
перегрузка

# Тяжелый хвост айтемов



# NETFLIX

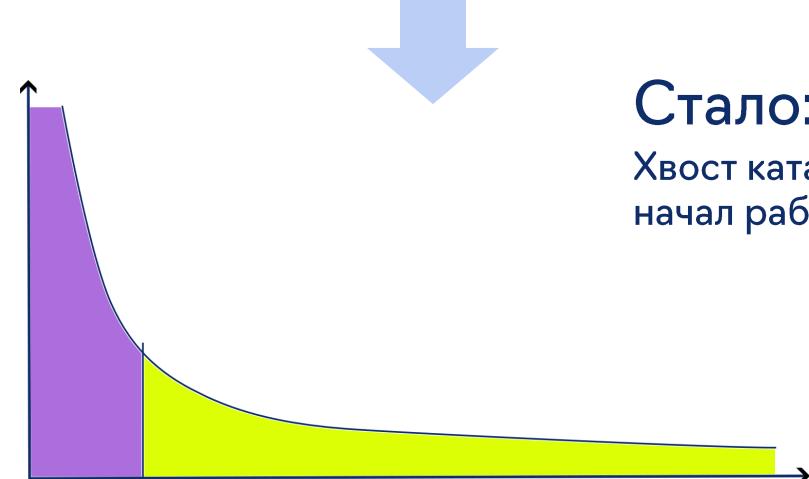
Рассылка DVD по почте:



Добавили персонализированные рекомендации на основе пользовательского фидбека (знаменитые 5-star ratings).



**БЫЛО:**  
Пользователи  
брали только  
новинки.



**Стало:**  
Хвост каталога  
начал работать.

# Кому и зачем нужны рекомендации

- Пользователю – найти интересный контент
- Автору – найти свою аудиторию (особенно новому)
- Платформе – зарабатывать

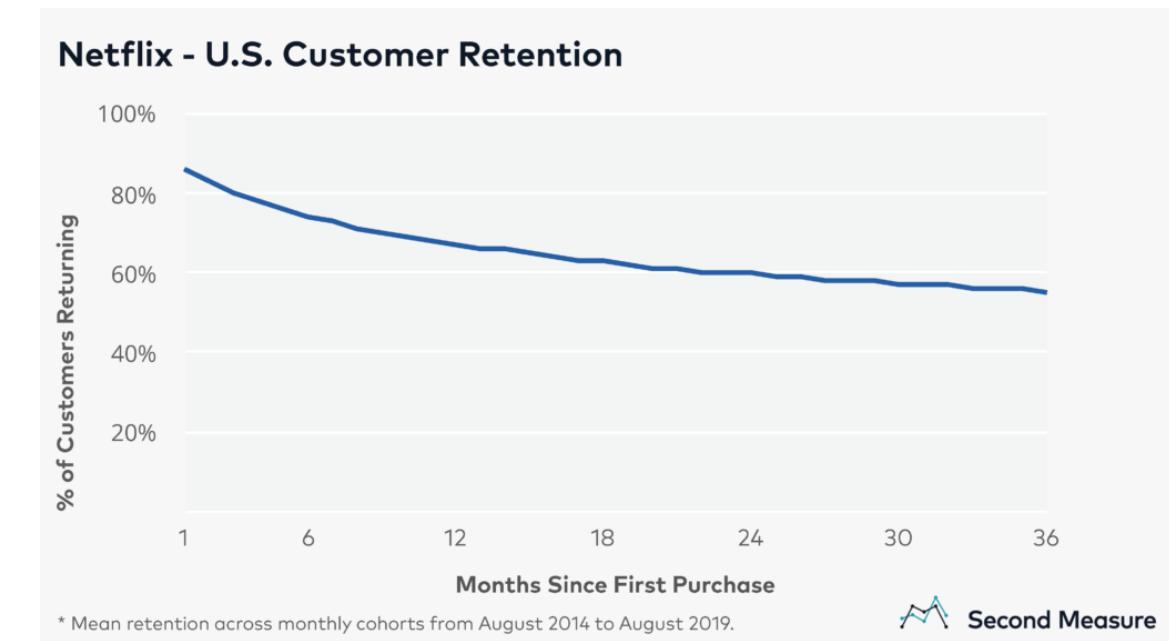
Рекомендации – это информационный фильтр, без которого цифровой мир не работает.



# Подписочные сервисы

Цель платформы:

- Удерживать текущих подписчиков (**retention**)
- Привлекать новых



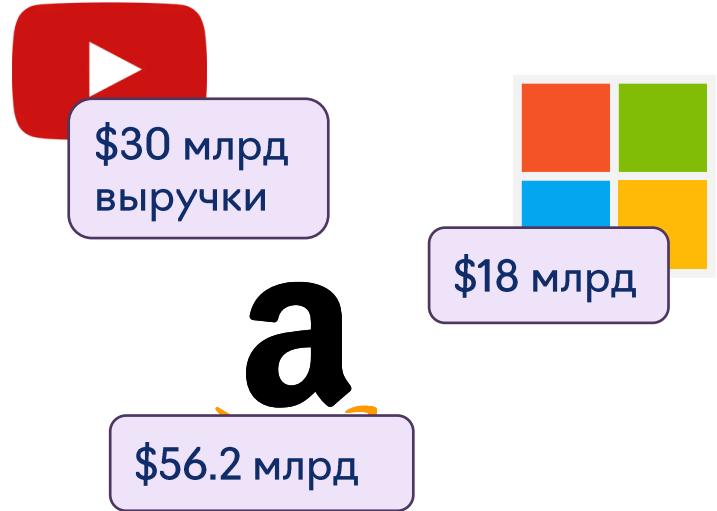
**Проблема холодного старта**  
(пользователей): трудно угадывать интересы новых пользователей.



# Монетизация через рекламу

YouTube:

- Во время просмотра видео, пользователю показывается реклама
- Чем больше watch time, тем больше показов рекламы



Больше активности → больше рекламы →  
больше доход.



Монетизация поиска: показы  
рекламных документов рядом  
с "органикой".

Google Search: \$198 млрд выручки

# Как устроены рекомендательные системы?

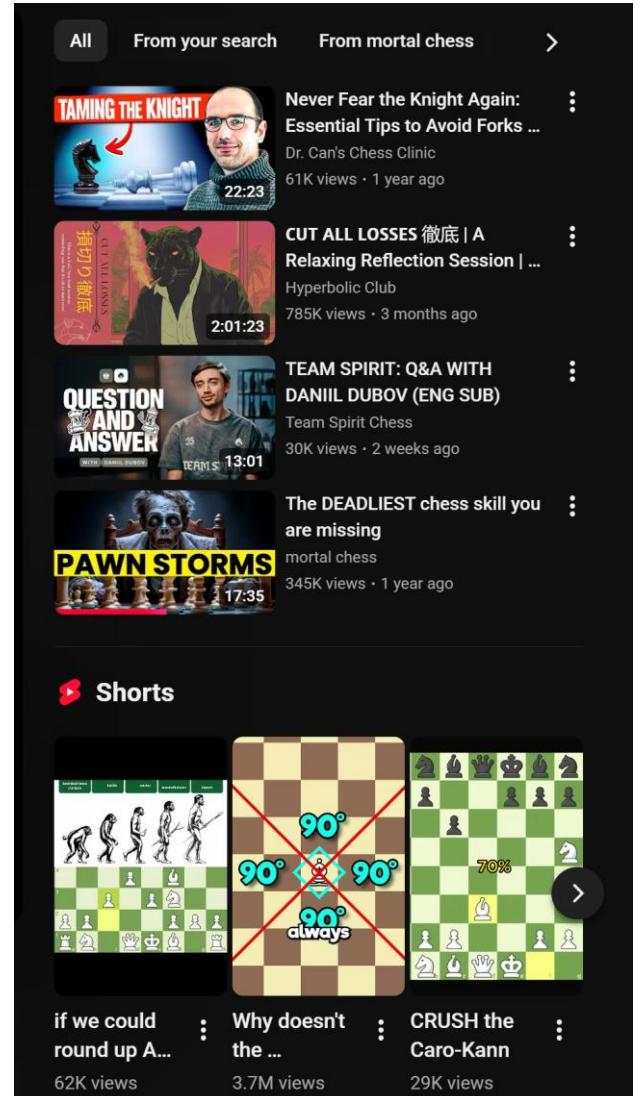
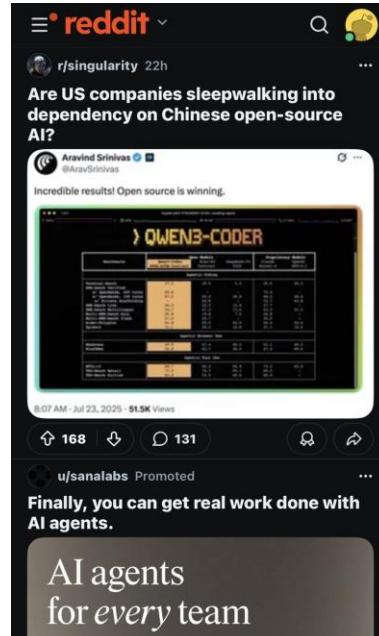
Часть 2



# Результат работы рекомендательной системы

- Следующий трек / видео / пост
- Лента новостей или рекомендаций
- Список похожих товаров
- Смесь разных форматов контента (и рекламы)

Интерфейсы разные, ранжирование  
одинаковое (почти).



# Главная формула



- Рексистема считает функцию  $f(u, c, i)$
- По значению  $f$  ранжирует весь каталог
- Вид  $f$  зависит от целей рексистемы:
  - Вероятность клика / лайка
  - Вероятность покупки, помноженная на цену
  - Ожидаемое время просмотра

# Что такое контекст?



Что является контекстом в:

- рекомендациях?
- поиске?
- рекомендациях похожих товаров?
- рекламе?

# Что такое контекст?



Что является контекстом в:

- рекомендациях? Время, устройство, браузер, поверхность, настройки
- поиске? Поисковый запрос
- рекомендациях похожих товаров? Товар, на карточке которого находимся
- рекламе? Веб-страница, на которую зашёл пользователь

# Скорость рекомендаций



Рекомендации должны формироваться за  
**десятки-сотни мс.**

# Многостадийность



Фокус на скорость и эффективность – эвристики и хитрые структуры данных (поиск ближайших соседей, обратный индекс), легкие модели

Тяжелая модель, использующая максимум информации



Многостадийность – это факторизация вычислений ради скорости.



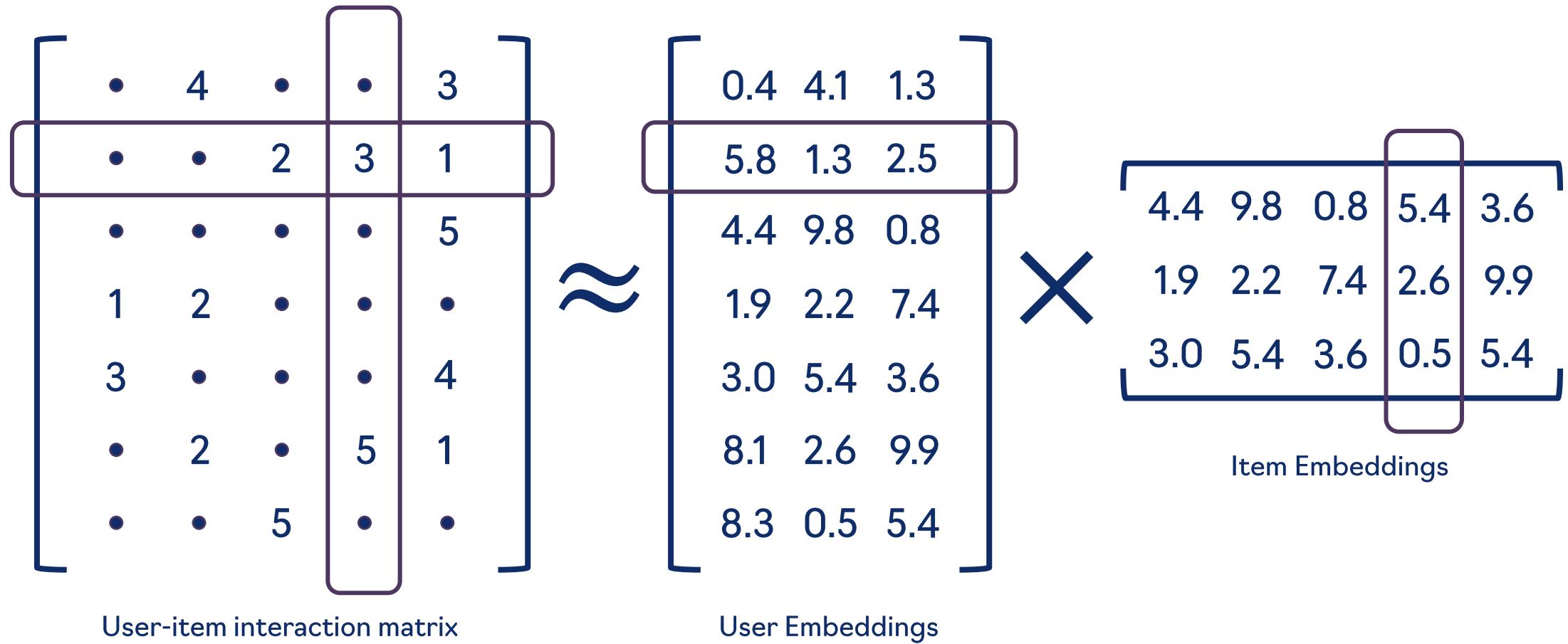
## Watch to watch next

Классический рекомендательный стек на примере YouTube до 2016 года – две стадии:

1. Генерация кандидатов: матричная факторизация
2. Ранжирование: линейная модель, предсказывающая expected watch time

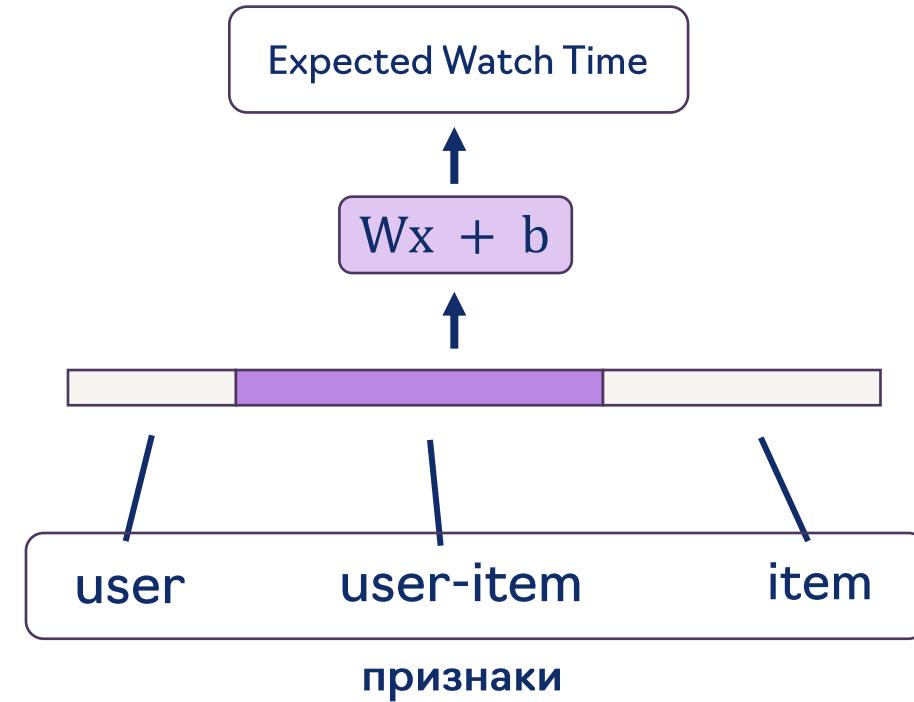
# Матричная факторизация

Представим user-item матрицу в виде произведения двух низкоранговых матриц:



# Линейная модель ранжирования

- Цель: показывать видео, которые будут смотреть дольше
- Модель предсказывает **expected watch time**
- Признаковое пространство:
  - признаки пользователя
  - признаки видео
  - user-item признаки (например, наличие подписки на автора)
- Обучение на показанных в прошлом рекомендациях



# Зачем нужны нейросети?

Часть 3



# Зачем нужны нейросети

- У нас триллионы user-item взаимодействий
  - Классическому ML столько не нужно
- В других областях DL уже победил
- В рекомендациях есть свои челленджи:
  - холодный старт и тяжелые хвосты
  - distribution drift



## Bitter lesson (Richard Sutton)

“The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin.”



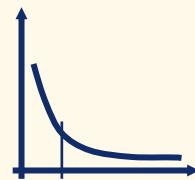
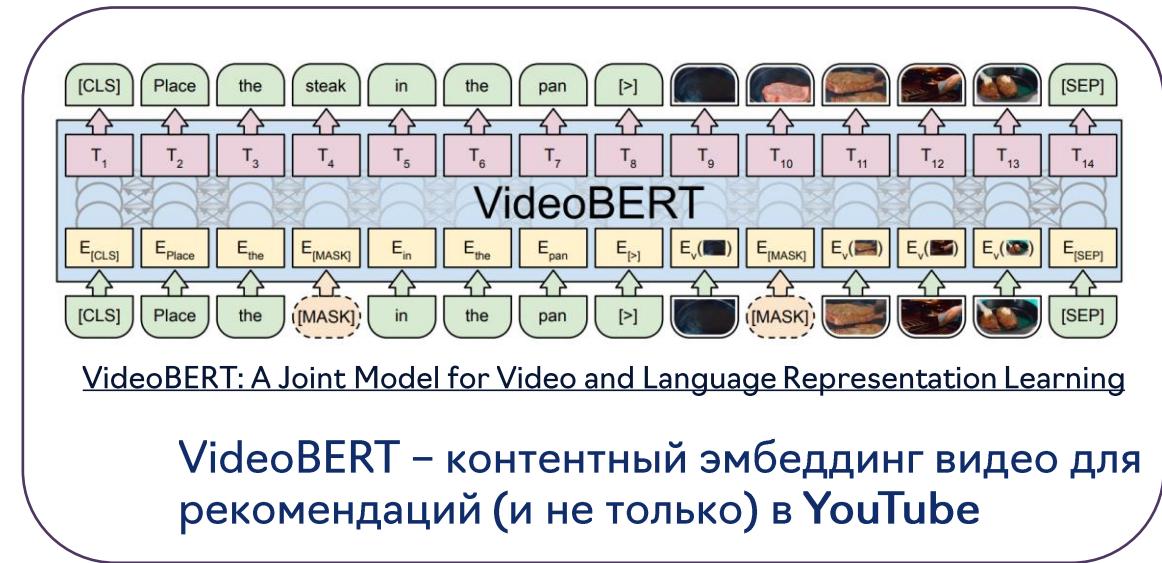
Scaling hypothesis – чем больше данных и сама модель, тем лучше результат.

# Холодный старт и тяжёлый хвост

- В новостях, рекламе маркетплейсах каталог постоянно обновляется
  - для новых айтемов нет фидбека
- Нужен **content understanding**:
  - видео/текст/картишка → эмбеддинг
- Тяжелый хвост:
  - популярные айтемы забивают обучение
  - у контентных признаков хвост легче, больше шанс вытянуть объекты из хвоста



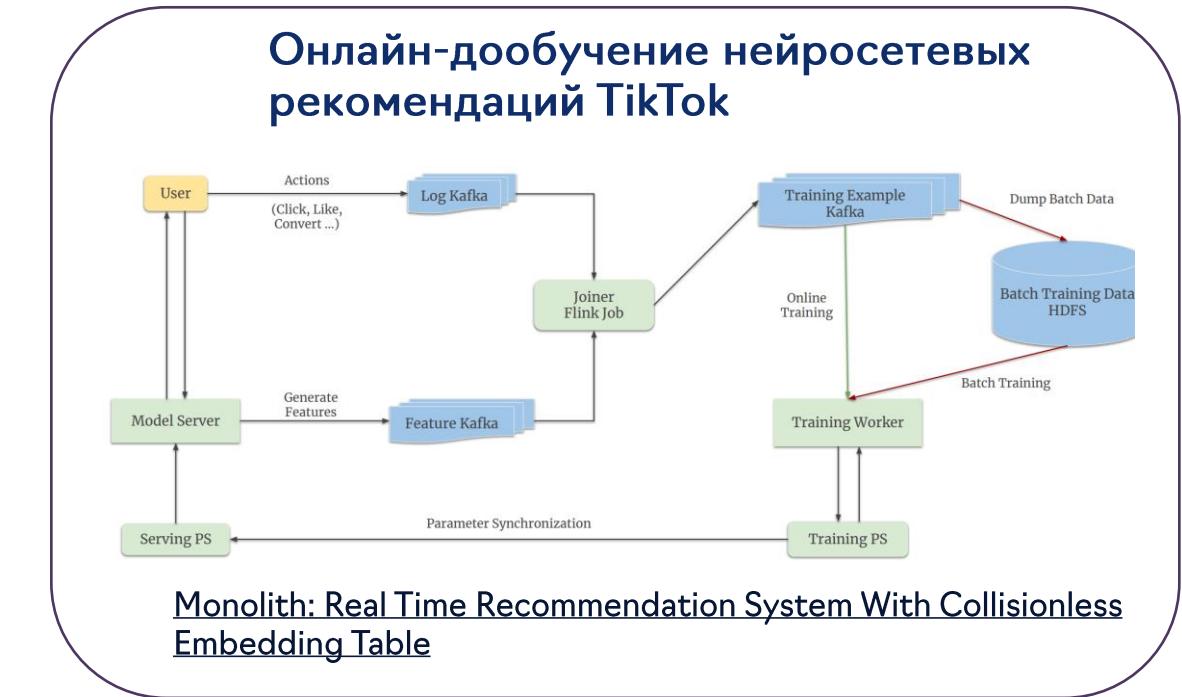
Холодный старт (айтемов): сложно рекомендовать новинки, для которых нет фидбека.



**Matthew Effect:** “the rich get richer and the poor get poorer.”  
Он же popularity bias.

# Distribution Drift

- Распределения постоянно меняются:
  - появляются новые тренды, обновляется каталог
  - меняются интересы пользователей
- Нужны алгоритмы, которые умеют быстро дообучаться на новых данных



## Feedback Loop



- Модель обучается на данных, которые сама же породила
- Усиливает свои bias'ы (например, popularity bias)
- Нужен exploration (bandits)

# Summary: зачем нам DL в рекомендациях

- **Масштаб** – хотим модели, которые скейлятся
- **Гибкость** – работать с разными типами данных (id, тексты, картинки, последовательности)
  - Улучшаем качество на тяжелом хвосте
- **Индуктивность** – решать проблему холодного старта
- **Адаптивность** – быстро подстраиваться под новые данные

# Архитектура нейросетевых рекомендаций

Часть 4



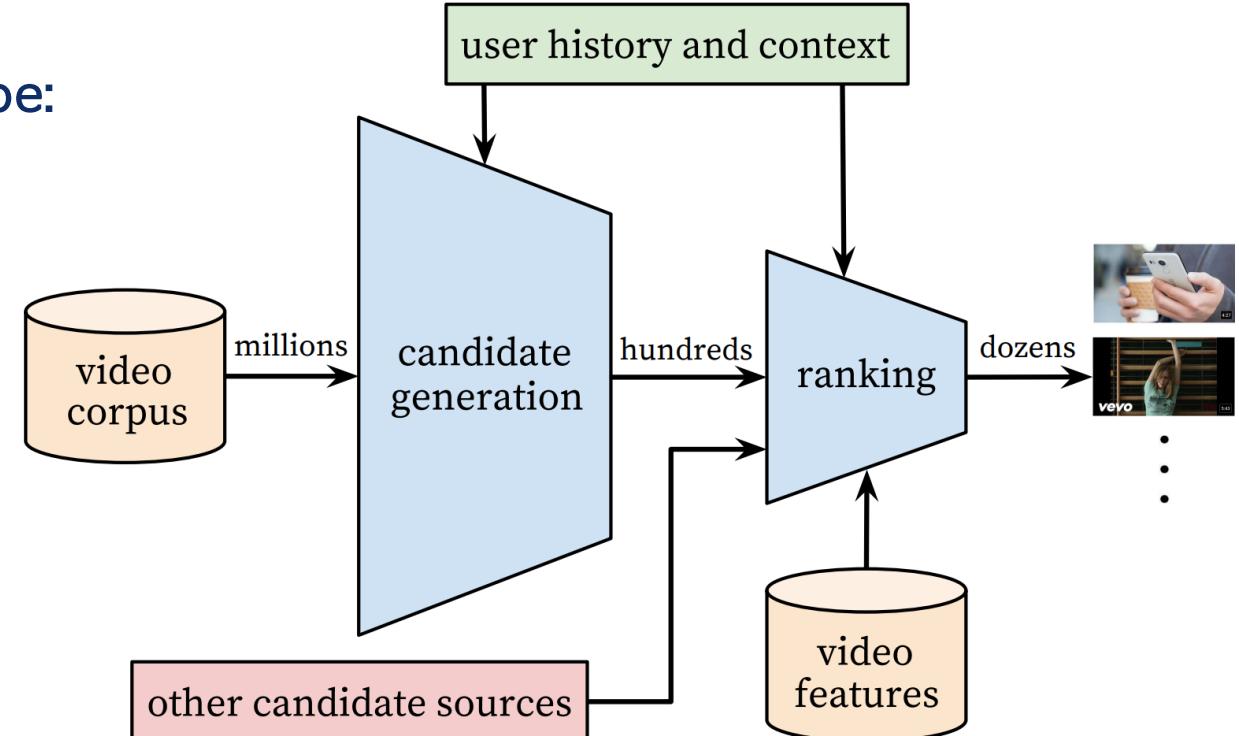
# YoutubeDNN

В 2016 году Google представили нейросетевые рекомендации в Youtube:

- **Двухбашенная нейросеть для генерации кандидатов**
- **Полносвязная нейросеть для ранжирования**

До этого:

- **Матричная факторизация** для генерации кандидатов
- **Линейная модель** для ранжирования



[Deep Neural Networks for YouTube Recommendations](#)

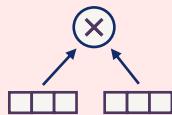
# Двухбашенные нейросети

Двухбашенные модели кодируют пользователей и айтемов в векторы:

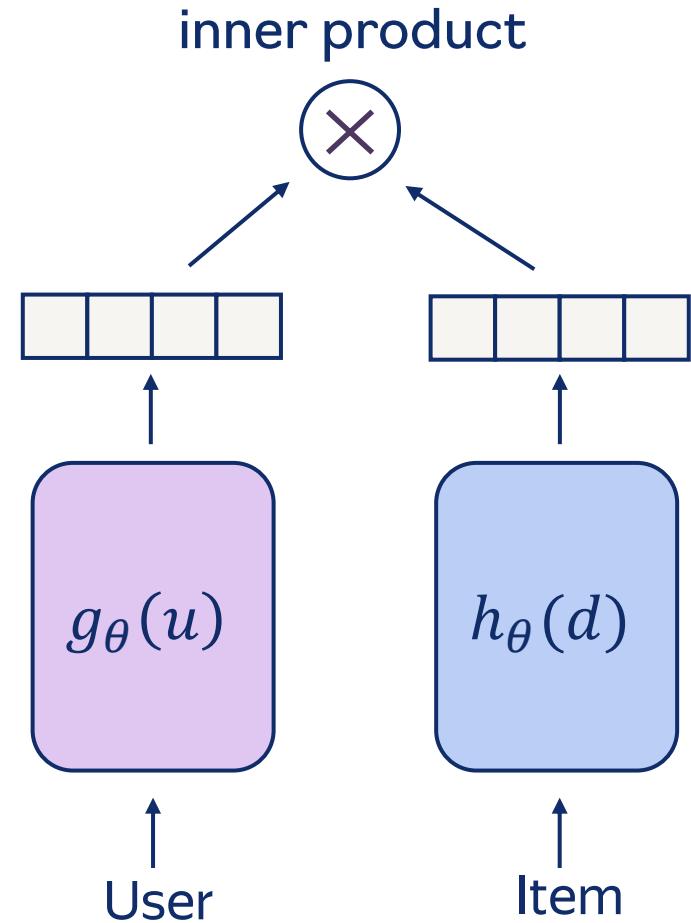
- Функция близости – скалярное произведение

$$f_{\theta}(u, d) = \langle g_{\theta}(u), h_{\theta}(d) \rangle$$

- Позволяют делать быстрый поиск соседей (ANNS), считать эмбеддинги оффлайн



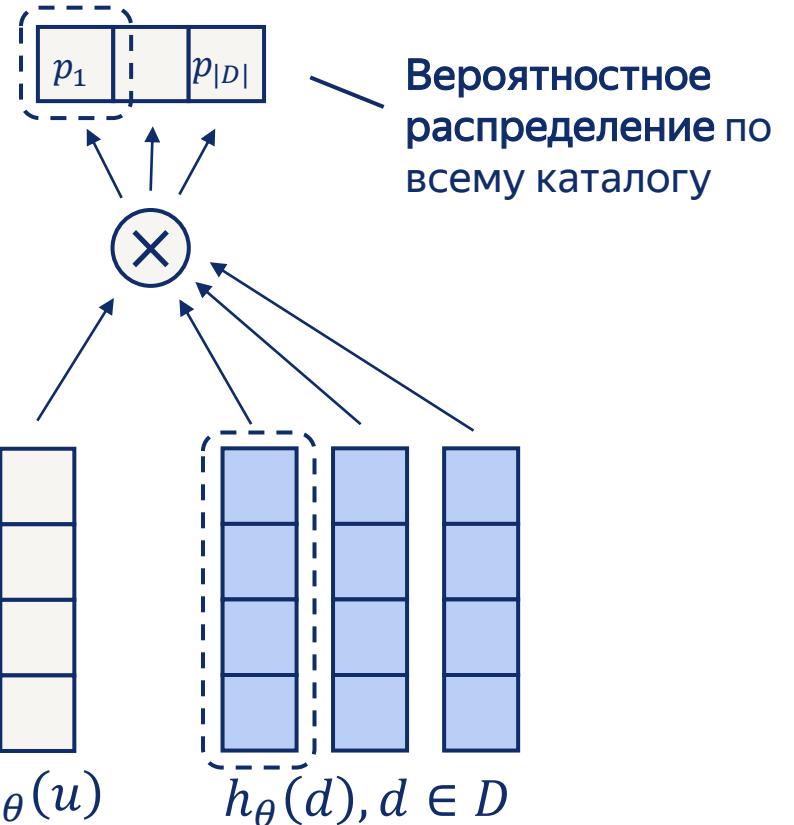
Матричная факторизация – это частный случай двухбашенной модели, в которой башни – это просто таблицы обучаемых векторов.



В 2013 году появился [DSSM](#) – первая двухбашенная нейросеть для информационного поиска.

# Обучение двухбашенных нейросетей

- **Next watch prediction:**
  - знаем, что пользователь посмотрел видео
  - хотим предсказать, что это за видео среди всего каталога
- **Это задача экстремальной классификации:**
  - Классы – это все видео в каталоге
- **“Разметка” для обучения приходит напрямую от пользователей (фидбек):**
  - Обучающий пример – пара (user, item)
- **Softmax по всему каталогу даёт глобальное сравнение всех айтемов**



$$\mathcal{L}_{\text{softmax}}(u, p) = -\log P_\theta(p | u) = -\log \frac{e^{f_\theta(u,p)}}{\sum_{d \in \mathcal{D}} e^{f_\theta(u,d)}}$$

# Проблема полного softmax

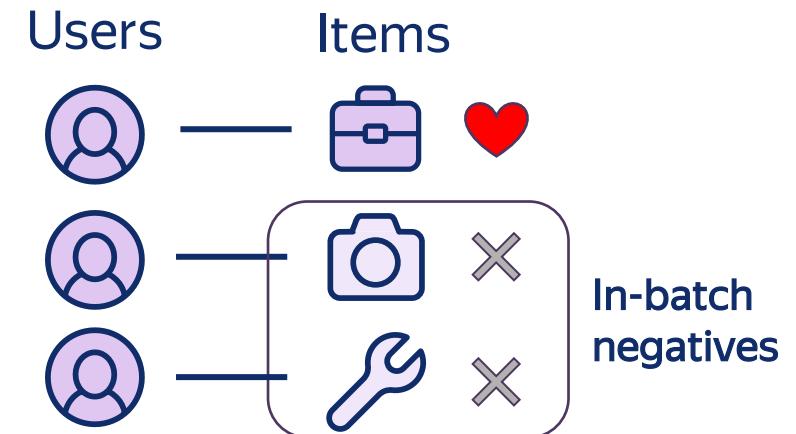
\*Correcting the LogQ Correction: Revisiting Sampled Softmax for Large-Scale Retrieval – наша статья с ACM RecSys'25 про улучшение logQ коррекции.

- В каталоге – миллионы / миллиарды айтемов
- Считать softmax по всему каталогу на каждом шаге обучения **невозможно**
- Решение – sampled softmax:
  - Берём настоящий позитив
  - Сэмплируем негативы

$$\mathcal{L}_{\text{sampled}}(u, p) = -\log \frac{e^{f_{\theta}(u,p)}}{e^{f_{\theta}(u,p)} + \sum_{d \sim Q} e^{f_{\theta}(u,d)}}$$

## Источники негативов:

- Равномерное сэмплирование по каталогу
- In-batch: позитивы других пользователей

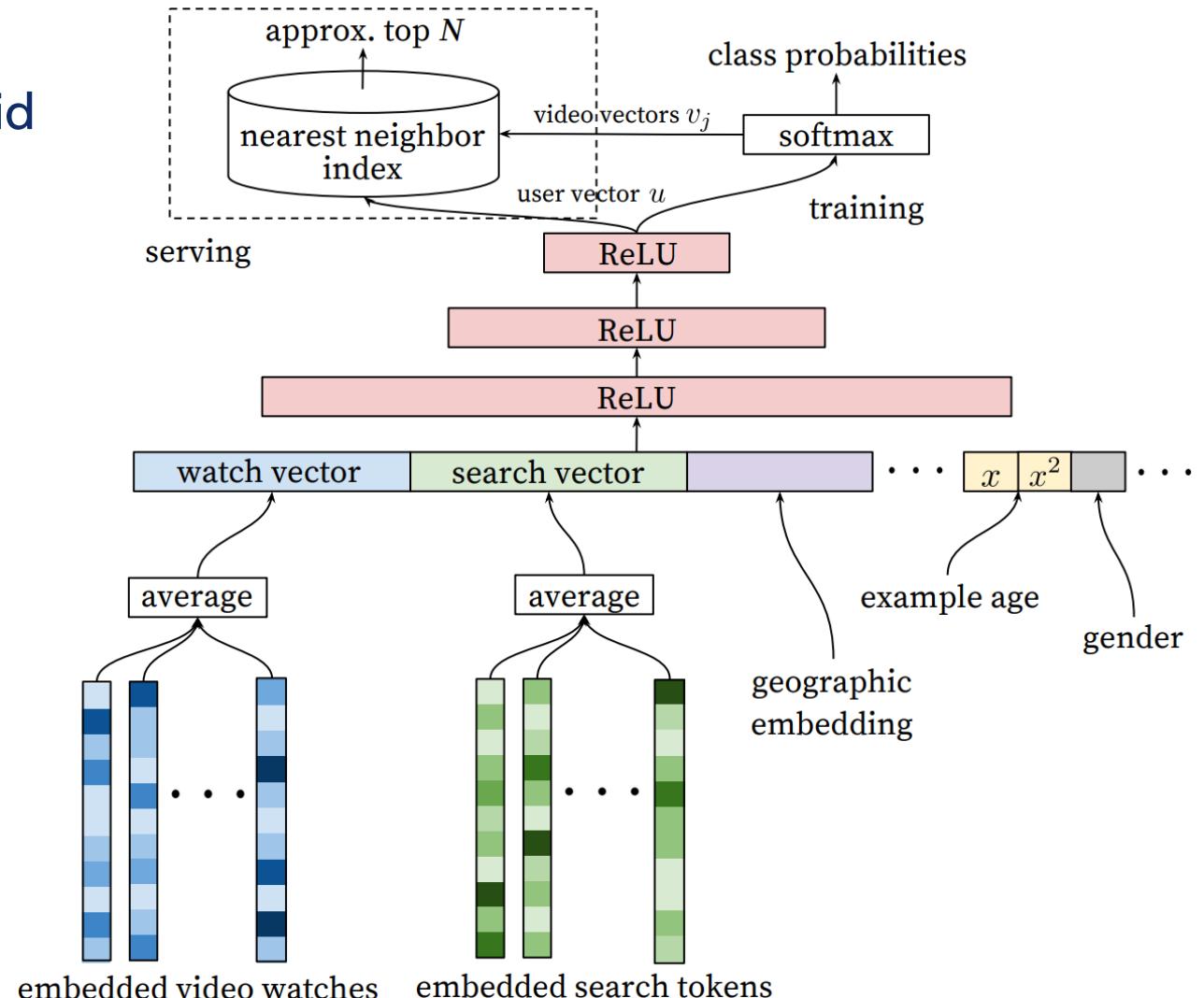
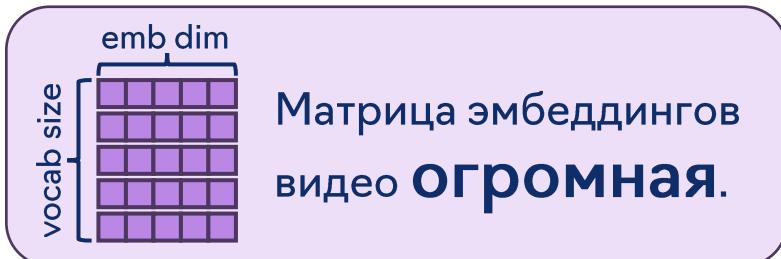


При in-batch сэмплировании популярные айтемы чаще появляются в негативах, поэтому модель начинает штрафовать за популярность. Фикс – logQ correction\*.

# YoutubeDNN: Retrieval

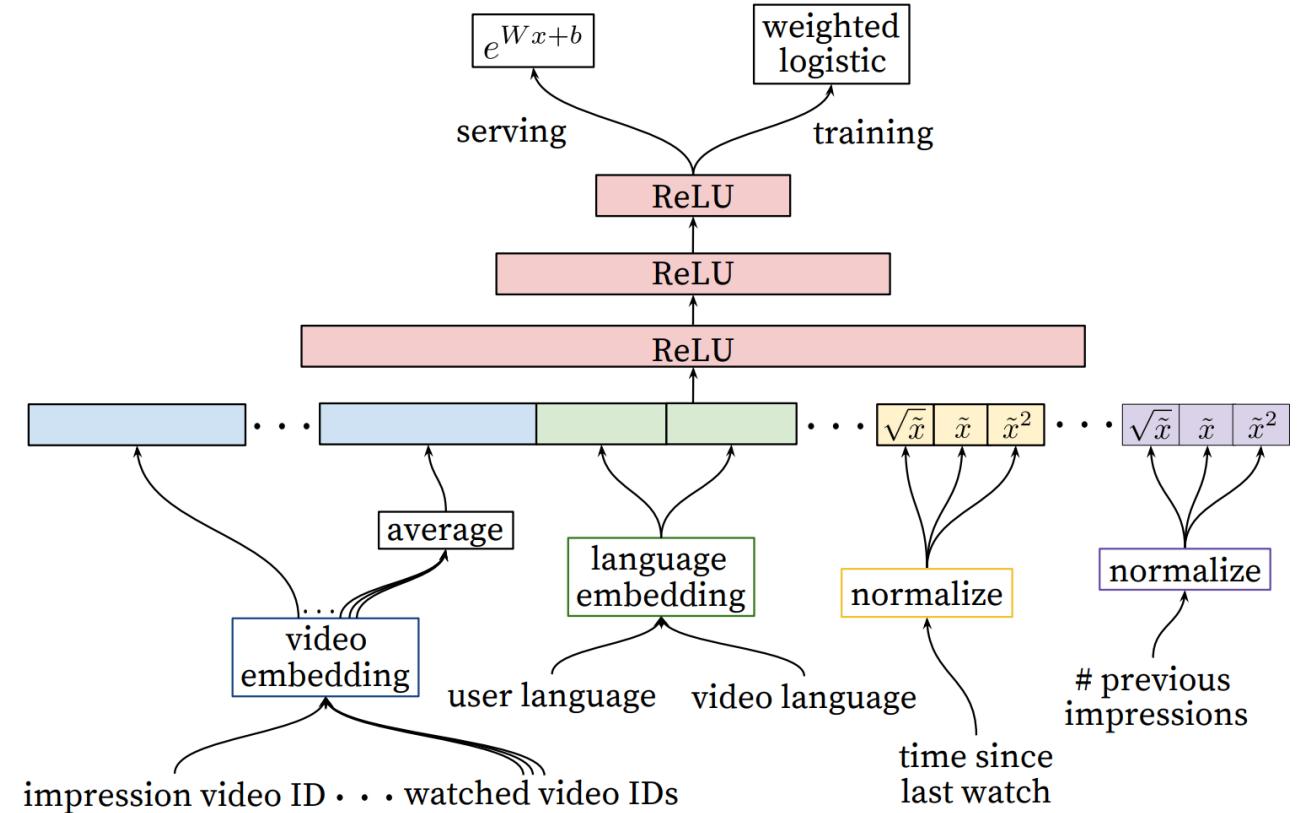
(генерация кандидатов)

- Обучаемые эмбеддинги для `video_id`
- Суммируем эмбеддинги последних просмотренных видео
- Добавляем признаки:
  - История поисковых запросов (мешок униграмм/биграмм)
  - Регион, устройство, язык, пол, и т.п.
- Выход: эмбеддинг пользователя

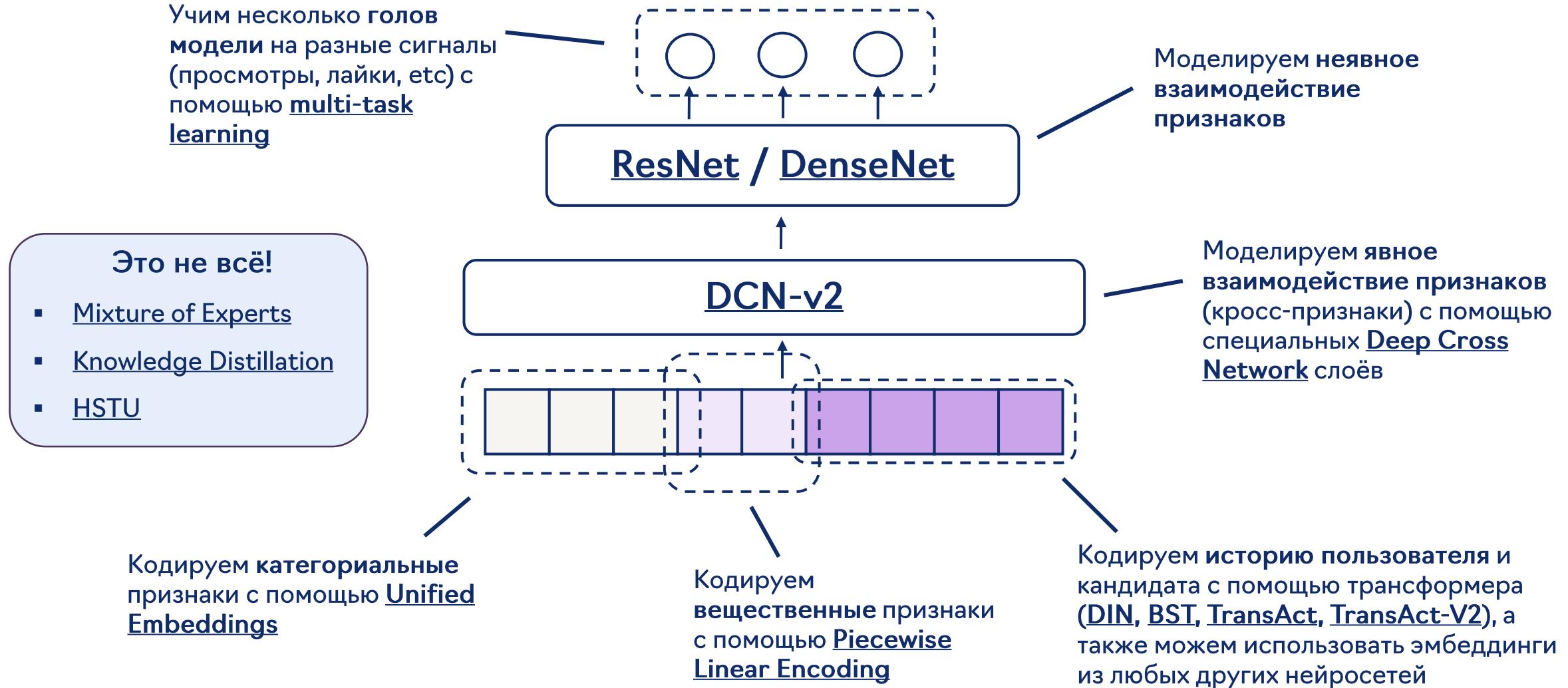


# YoutubeDNN: Ranking

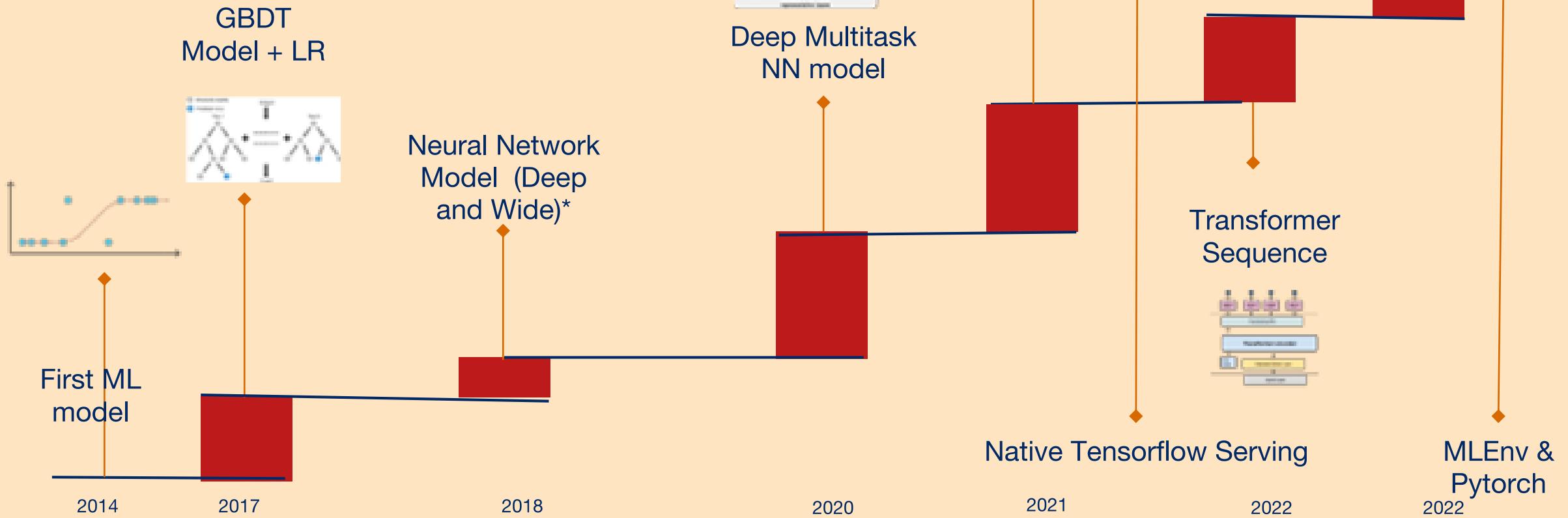
- Вместо линейной модели – MLP
- На ранжировании:
  - Меньше кандидатов → можно использовать больше признаков
- Раннее связывание user x item:
  - User-item счётчики (“сколько раз этот пользователь лайкал этого автора”)



# Эволюция нейросетевого ранжирования



# Pinterest Ranking Evolution



# Масштабирование рексистем

- **Матрицы эмбеддингов:** от миллионов до триллионов параметров
- **Датасеты:** до сотен миллиардов примеров
- **Ранжирование:** тысячи признаков, десятки миллионов параметров в нейросетях

Inductive bias  
мешает

Исторически (до 2024-го года) трансформеры в рексистемах были маленькие:

- <2M параметров
- Короткое контекстное окно: десятки-сотни токенов



# **Frontier:**

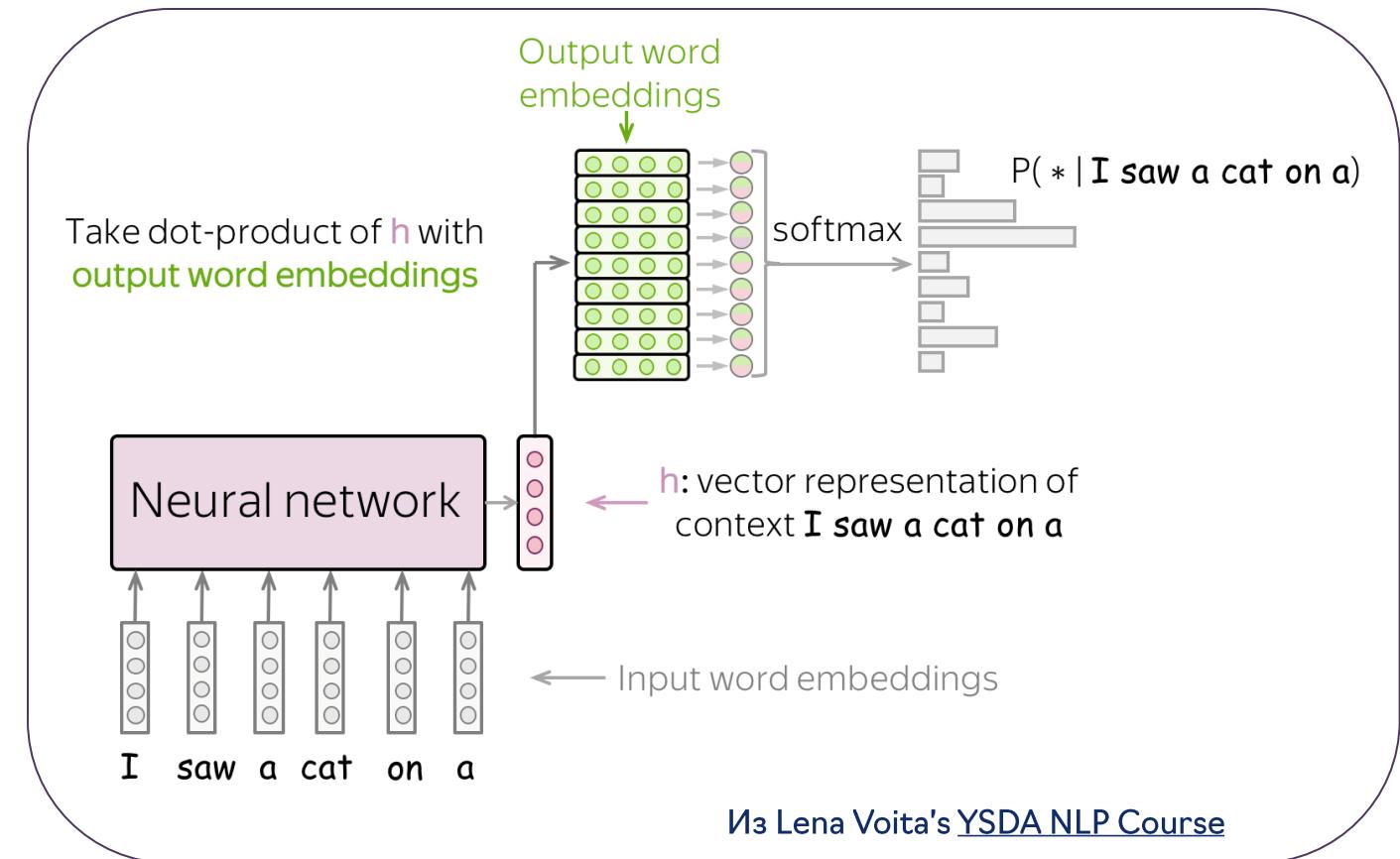
трансформеры, масштабирование, RL, генеративные  
рекомендации, разговорные рексистемы, semantic IDs

## **Часть 5**



# Large Language Models

- LLM учатся предсказывать следующее слово
- Extreme multi-task learning: грамматика, факты, логика, математика, здравый смысл, etc
- Архитектурно, LLM – это двухбашенная модель:
  - Левая башня: трансформер над текущим контекстом
  - Правая: матрица эмбеддингов токенов
  - Softmax по словарю

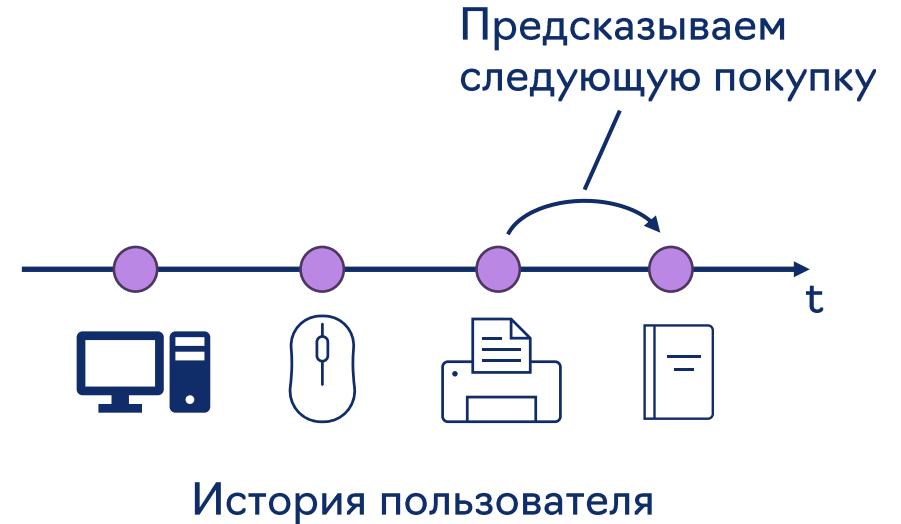


# Next Item Prediction

Заменим слова на действия пользователей:

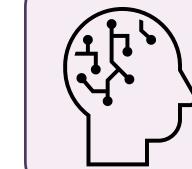
- Было (NLP): “Я пошёл в [token1] [token2] ...”
- стало (RecSys): “Пользователь слушал [track1] [track2] ...”

С помощью задачи  
**предсказания следующего айтема**  
обучаем генеративную модель пользователя.



В отличие от NLP, пространство токенов очень большое:

- NLP:  $10^4 - 10^5$ , RecSys:  $10^6 - 10^9$  токенов
- больше похоже на генерацию следующего кадра видео, чем на текст



Это модель мира! В которой мир – это пользователь.

# Как использовать NIP модель на практике

## 1. Генерация кандидатов:

- Считаем эмбеддинг пользователя через трансформер
- Ищем top-K айтемов по скалярному произведению (MIPS + ANNS)

## 2. Ранжирование:

- Эмбеддинги пользователей и айтемов – фичи в ранжирующей модели

## 3. Pre-train → fine-tune:

- Обучаем на next-item-prediction
- Дообучаем под конкретную задачу (например, ранжирование)

### Пример

История пользователя:  
8192 прослушиваний

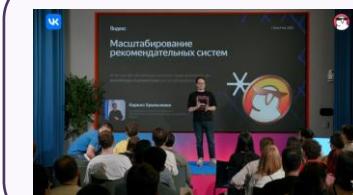
### Внедрили модель в Яндекс Музыку:

- Предобучаем на модифицированный next-item-prediction
- Дообучаем на ранжирование
- Используем как признак в ранжирующем бустинге

Суммарное время  
прослушивания: **+2.26%**

Вероятность  
лайка: **+6.37%**

[Scaling Recommender Transformers to One Billion Parameters](#) – наша статья, принятая на KDD'26.



[Масштабирование  
рекомендательных  
систем, Data Fest  
2025](#)

# Трансформеры в Яндекс Музыке



Уже есть и новые модели, смотрите  
[выступление Петра Зайделя на PML Conf.](#)

# Трансформеры в Яндекс Музыке

Внедрение	Длина истории	Конфигурация	Параметры	TLT	Like Likelihood
Offline V1	512	L6 H512	18.9M	+0.52%	+1.11%
Offline V2	1024	L6 H512	18.9M	+1.00%	+0.73%
Offline V3	1024	L6 H512	18.9M	+0.73%	+5.00%
Real-time V1	1024	L4 H256	3.2M	+0.32%	+1.38%
<b>Offline V4 (ARGUS)</b>	<b>8192</b>	<b>L10 H1024</b>	<b>126.0M</b>	<b>+2.26%</b>	<b>+6.37%</b>

# Beyond Short-term Optimization

- Реальные KPI платформы:
  - долгосрочная вовлеченность
  - удержание, подписки, LTV
- Типичная модель оптимизирует:
  - Вероятность просмотра / лайка на следующем видео



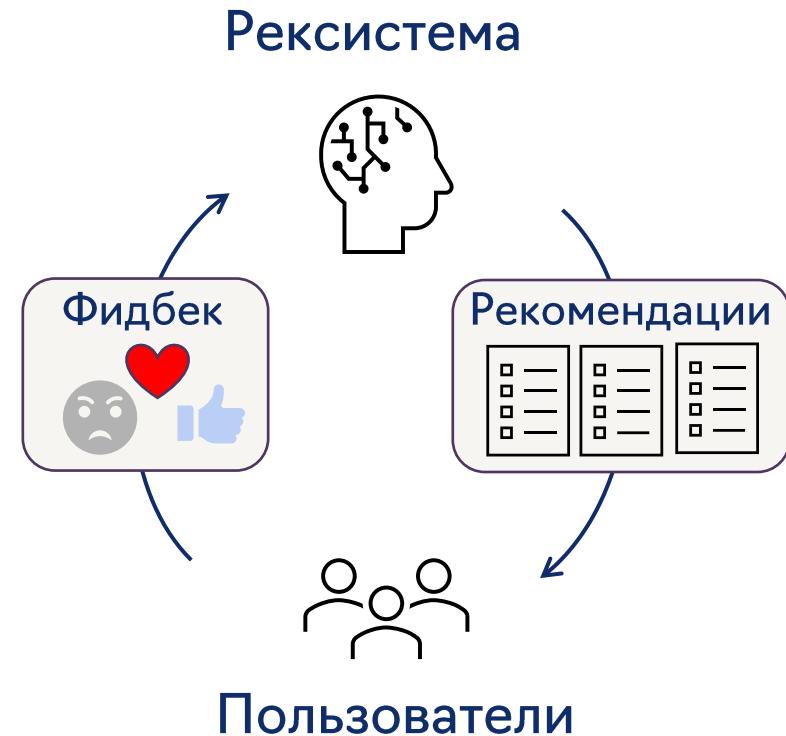
Пользователь смотрит TikTok 3 часа подряд – это хорошо или плохо?

**Filter bubble** – алгоритм сильно ограничивает потребляемую пользователем информацию, держит его “в пузыре”.

# Рекомендации как RL

- Агент: рексистема
- Действия: показы рекомендаций
- Окружение: пользователи
- Награда: клики, покупки, подписки, просмотры

LTV (lifetime value) – кумулятивная награда из RL.



# Рекомендации – это RL, но очень сложный RL

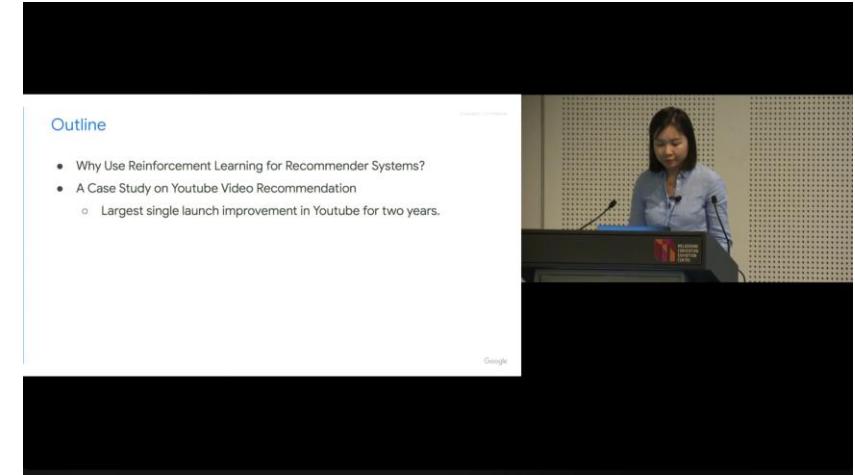
- Гигантские пространства состояний и действий
- Нет хорошего симулятора пользователей
- Можем учиться только off-policy – на залогированных данных
- На практике настоящий RL почти никто не использует



# Top-K Off-Policy Correction for a REINFORCE Recommender System

- В Youtube использовали RNN\* над историей просмотров
- Стали максимизировать **watch time** на горизонте 4-10 часов
- Использовали REINFORCE:
  - взвешенный sampled softmax (на основе долгосрочного сигнала)
  - off-policy correction

**Результат:** самое большое единоразовое улучшение рекомендаций за два года.



[WSDM 2019, Reinforcement Learning of Recommender Systems: A Case Study on Youtube](#)



**REINFORCE:** увеличиваем вероятности действий, участвующих в “хороших” траекториях.

\*[Latent Cross: Making Use of Context in Recurrent Recommender Systems](#)

# OneRec: одностадийная генеративная рекомендательная система

- Заменили весь рекомендательный стек на одну end-to-end модель
- Используют одну генеративную модель, которая:
  - Генерирует мини-сессию: несколько следующих видеороликов сразу
  - Использует только историю пользователя
  - Кодирует (и декодирует) айтемы через семантические идентификаторы
- Есть reward model + preference alignment по аналогии с LLM (аналог DPO)

## Пример рекомендательных метрик

OneRec-V1 A/B testing results		
Model	Total Watch Time	Average View Duration
OneRec-0.1B	+0.57%	+4.26%
OneRec-1B	+1.21%	+5.01%
OneRec-1B+IPA	+1.68%	+6.56%

## OneRec-V2 Technical Report

Scenarios	Online Metrics	OneRec-V2
Kuaishou	App Stay Time	+0.467%
	LT7	+0.069%
	Watch Time	+1.367%
	Video View	+0.331%
	Like	+3.924%
	Follow	+4.730%
	Comment	+5.394%
	Collect	+2.112%
	Forward	+3.183%

[OneRec: Unifying Retrieve and Rank with Generative Recommender and Iterative Preference Alignment](#)

# Recommender Systems as Manipulation Engines

- Хорошо работающий RecSys RL = сильное влияние на будущее пользователей
  - Влияем на то, что пользователь будет смотреть / покупать / читать – сегодня, завтра, через месяц, в ближайший год
- Метрика задаётся платформой и одинакова для всех
- Эхо-камеры, радикализация, информационные пузыри

**Echo chamber** – ситуация, в которой пользователи потребляют только информацию, подтверждающую их взгляды



У разных людей разные функции полезности. Нужно давать пользователю контроль над рекомендациями.

# Conversational Recommender Systems

- Рексистема, с которой можно поговорить:
  - Пользователь явно формулирует свои пожелания
  - Может исправлять, уточнять, задавать ограничения
- В поиске уже есть похожая эволюция:
  - 8 blue links → AI-generated answers
- Универсальные ассистенты (типа project Astra) должны уметь и общаться, и рекомендовать



[Ed Chi - Google Gemini Era: Bringing AI to Universal Assistant and the Real World](#)

Каждый год во время своих выступлений хвалится, что у него на телефоне уже есть project Astra :)

# P5: Recommendation as Language Processing

- Представим всё как текст:
  - User-item взаимодействия, метаданные
  - Решаем все задачи text-to-text с помощью encoder-decoder (T5)
- Пользователей и айтемы кодируем как строки "user\_23", "item\_7133":
  - Токенизация: "user", "\_", "23" / "item", "\_", "7133"
  - **Vocabulary gap:** пробовали кодировать каждого пользователя и айтем отдельным токеном, работает плохо

## Обучающий сэмпл

- **Input:** "инструкция + текстовые поля про юзера, айтем, историю, кандидатов..."
- **Target:** нужный ответ в текстовом виде (рейтинг, id айтема, объяснение, summary)

## Instruction-based prompting

Задача задаётся через текст:

- "Predict the star rating..."
- "Recommend the next item..."
- "Explain why the user likes..."

## Sequential Recommendation

I find the purchase history list of user\_15466:

4110 -> 4467 -> 4468 -> 4472

I wonder what is the next item to recommend to the user. Can you help me decide?

## Rating Prediction

What star rating do you think user\_23 will give item\_7391?

## Explanation Generation

Help Hong "Old boy" generate a 5-star explanation about this product:  
OtterBox Defender Case for iPhone 3G, 3GS (Black) [Retail Packaging]

## Review Summarization

Give a short sentence describing the following product review from Mom of 3 yo girl:  
First it came with the packaging open and then as soon as my son took it out it was so easily broken. Hopefully a little glue will fix it.

## Direct Recommendation

Pick the most suitable item from the following list and recommend to user\_250 : \n 4915 , 1823 , 3112 , 3821 , 3773 , 520 , 7384 ,  
7469 , 9318 , 3876 , 1143 , 789 , 595 , 3824 , 3587 , 10396 , 2766 ,  
7498 , 2490 , 3232 , 9711 , 2975 , 1427 , 9923 , 3097 , 3594 ,  
6469 , 9460 , 6956 , 9154

## Multi-task Pretraining with Personalized Prompt Collection

## Zero-shot Generalization to New Product & Personalized Prompt

Predict user\_14456 's preference about the new product  
( 1 being lowest and 5 being highest ) : \n title : Hugg-A-Moon  
\n price : 13.22 \n brand : Hugg-A-Planet

**P5**

1581

5.0

you can protect your precious  
iphone more safe

broke immediately

520

4.7

## Проблемы Р5

- Нет связи между токенами похожих айтемов
- Нет иерархии: префикс не обозначает класс или кластер

## Проблемы P5

- Нет связи между токенами похожихアイテムов
- Нет иерархии: префикс не обозначает класс или кластер

**Naively Structured String Identifiers** unstructured identifiers, i.e., arbitrary unique integers, as tokenizable strings. We also consider an **ostensibly absurd** approach that treats naively structured identifiers.

Авторы статьи  
“Transformer Memory  
as a Differentiable  
Search Index”

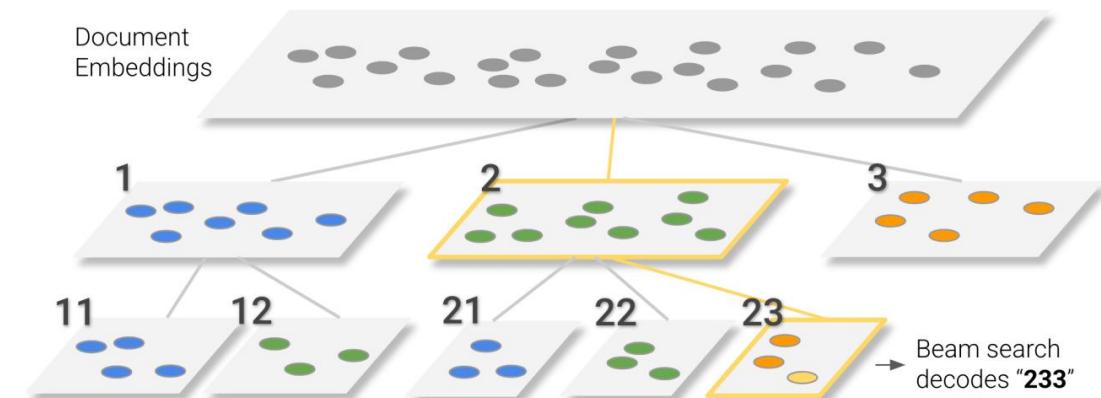
# Transformer Memory as a Differentiable Search Index

- **Задача индексации:** запомнить весь поисковый индекс в параметрах модели
- **Обучение:**
  - **вход:** содержимое документа
  - **выход:** последовательность токенов docid
- **Вторая задача – generative retrieval:**
  - **вход:** текстовый запрос (вопрос, ключевые слова)
  - **выход:** последовательность токенов docid, генерируем авторегрессивно (beam search)



Opinion paper "Rethinking Search: Making Domain Experts out of Dilettantes"

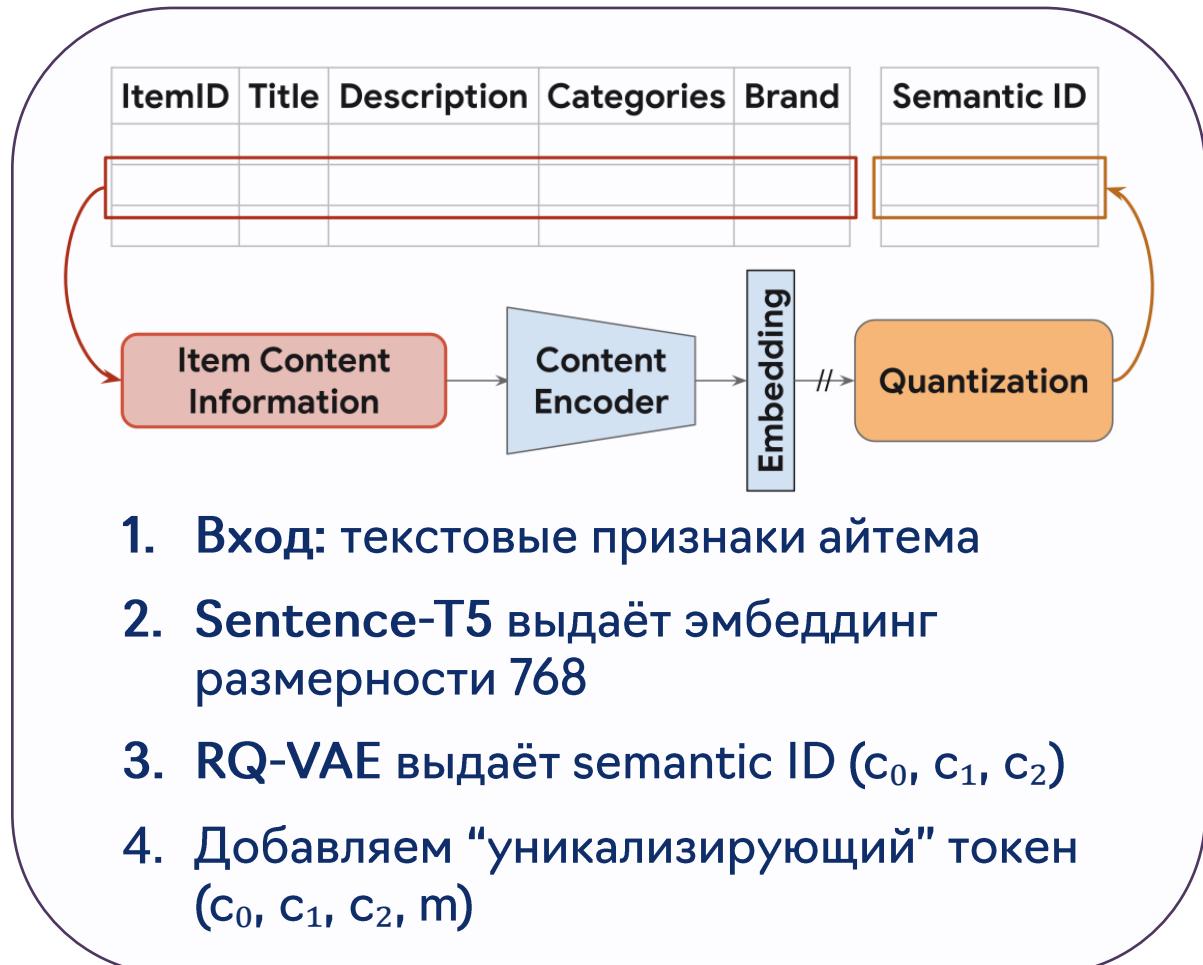
- Можно ли превратить языковые модели в полноценные информационные системы?
- Хотим избавиться от поисковых индексов и двухбашенных нейросетей



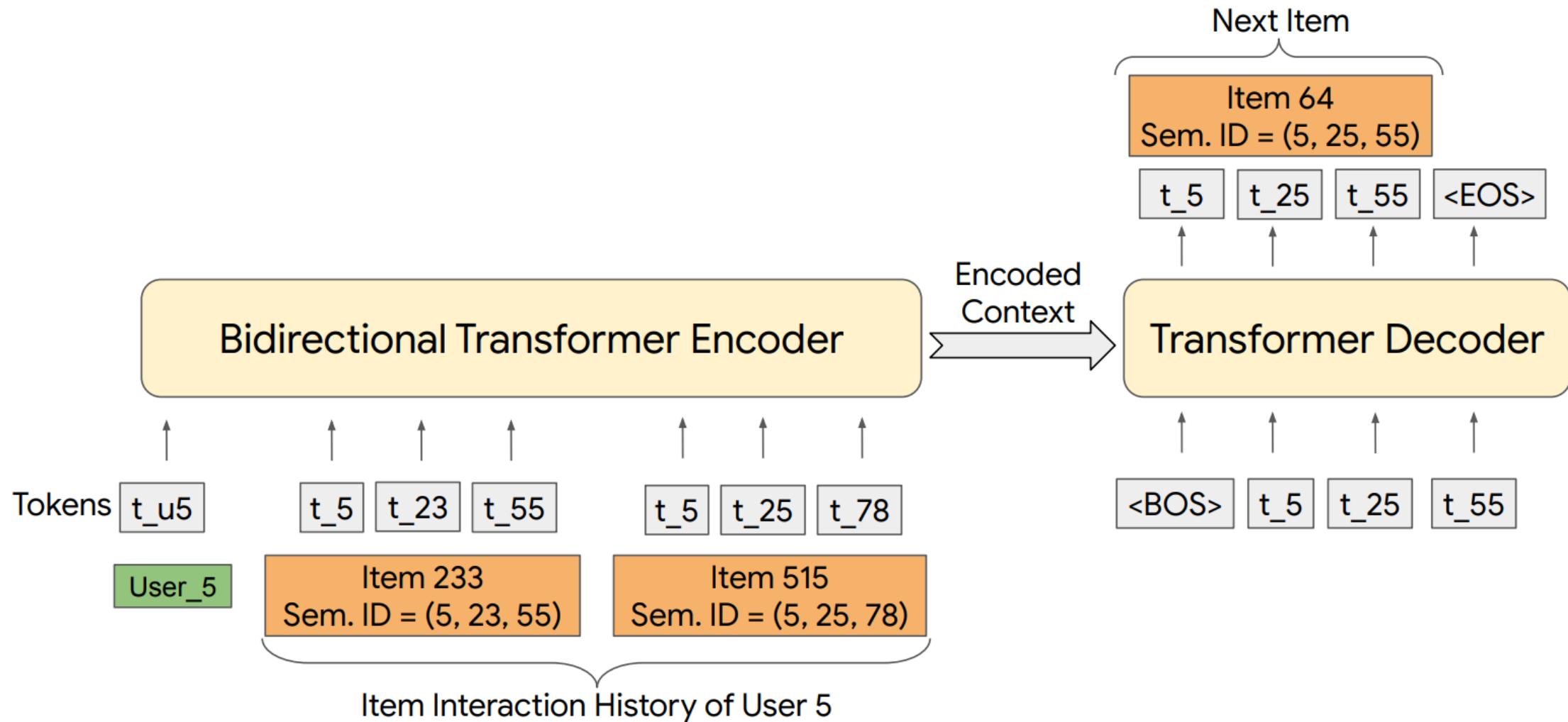
Иерархический k-means для получения семантических идентификаторов документов

# TIGER: Transformer Index for Generative Recommenders

- Среди авторов – первый автор DSI (Yi Tay)
- **Semantic IDs (SIDs):**
  - обучили RQ-VAE над контентными векторами айтемов
  - У похожих айтемов – похожие префиксы (SID)
- **Generative retrieval:**
  - по истории пользователя предсказывают SID следующего айтема
  - SIDs помогают с холодным стартом и тяжелым хвостом айтемов

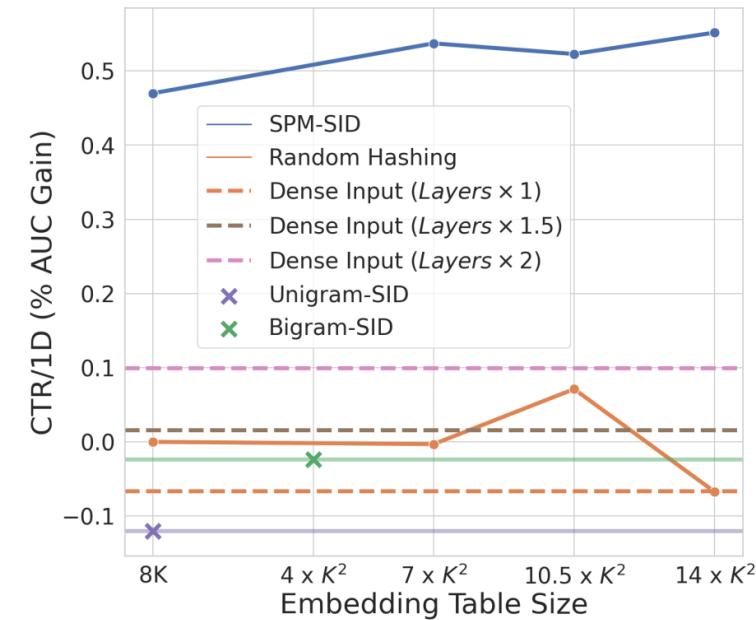


# TIGER: Encoder-Decoder Architecture



# Semantic IDs in YouTube Ranking

- Заменили обычный `video_id` эмбеддинг на `semantic IDs`:
  - Обучили RQ-VAE поверх контентных эмбеддингов видео (VideoBERT)
- Результат: значительный прирост качества на холодном срезе



Если заменить `video_id` эмбеддинг на контентный, качество падает – теряем **полезную меморизацию**.

[Better Generalization with Semantic IDs: A Case Study in Ranking for Recommendations](#)

# PLUM: адаптация Gemini под YouTube

- Цель: дообучить большую LLM (Gemini) для задач рекомендаций
- Три стадии:
  - Семантическая токенизация – обучаем RQ-VAE поверх эмбеддингов видео, получаем SIDs
  - Continuous pre-training – учим модель понимать и естественный язык, и SIDs
  - Supervised fine-tuning – обучаем generative retrieval
- Результат: самое большое улучшение рекомендаций за несколько лет

Не вчитывайтесь :)

## Semantic IDs v2

- Более мощный мультимодальный контентный эмбед
- Дополнительный contrastive loss на основе поведенческих данных задача при обучении RQ-VAE
- Multi-resolution codebook & progressive masking

[PLUM: Adapting Pre-trained Language Models for Industrial-scale Generative Recommendations](#)

# PLUM: Continuous Pre-training

- Смешиваем два типа данных:
  - История поведения пользователей
  - Корпус метаданных видео
- Модель учится:
  - По истории предсказывать SID следующего просмотра
  - По SID восстанавливать метаданные  
Смешивают данные 50-50
- Сохраняем языковые способности и добавляем знание рекомендательного домена

---

## **Example user behavior training data**

---

wh = <sid\_1> <channel\_name> <watch\_ratio> <watch\_time>  
<hours\_since\_final\_watch> <sid\_2> <channel\_name> ... || <sid\_n>

---

## **SID + video title**

---

Video <sid> has title (en): <video\_title>

---

## **SID + video topics**

---

The topics in video <sid> are: <topics>

---

## **In-context few-shot learning**

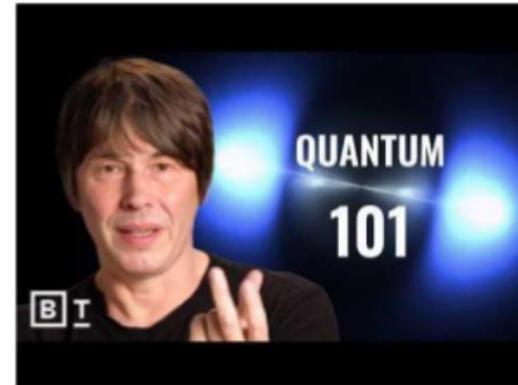
После СРТ стадии модель всё еще умеет общаться на естественном языке, а также отвечает на вопросы про SIDs.

# PLUM: In-Context Few-Shot Learning

(a) Example 1



(1/3) B6t2IY9sUxA  
**Natural Remedies for Hypothyroidism and Hashimoto's Disease**  
Dr. Josh Axe  
556827 views



(2/3) BHEhxPuMmQI  
**Physicist Brian Cox explains quantum physics in 22 minutes**  
Big Think  
2365348 views



(3/3) iNyUmbmQQZg  
**Is it normal to talk to yourself?**  
TED-Ed  
8682099 views

## Video Thumbnail

### Few-shot Input

The video A1991 ... H10 is about health and pain management.  
The video A364 ... H37 is about the basics of quantum physics.  
The video A37 ... H25 is about

### LLM-Initialized CPT Output

psychology and mind

### Randomly-Initialized CPT Output

100% video A252 H7

# PLUM: In-Context Few-Shot Learning

(b) Example 2



## Video Thumbnail

### Few-shot Input

Video A37 ... H41 answers the question: do you really need 8 hours of sleep?  
Video A1926 ... H8 answers the question: how buy-now pay-later loan business makes profit?  
Video A1882 ... H32 answers the question:

### LLM-Initialized CPT Output

the impact of language in your life.

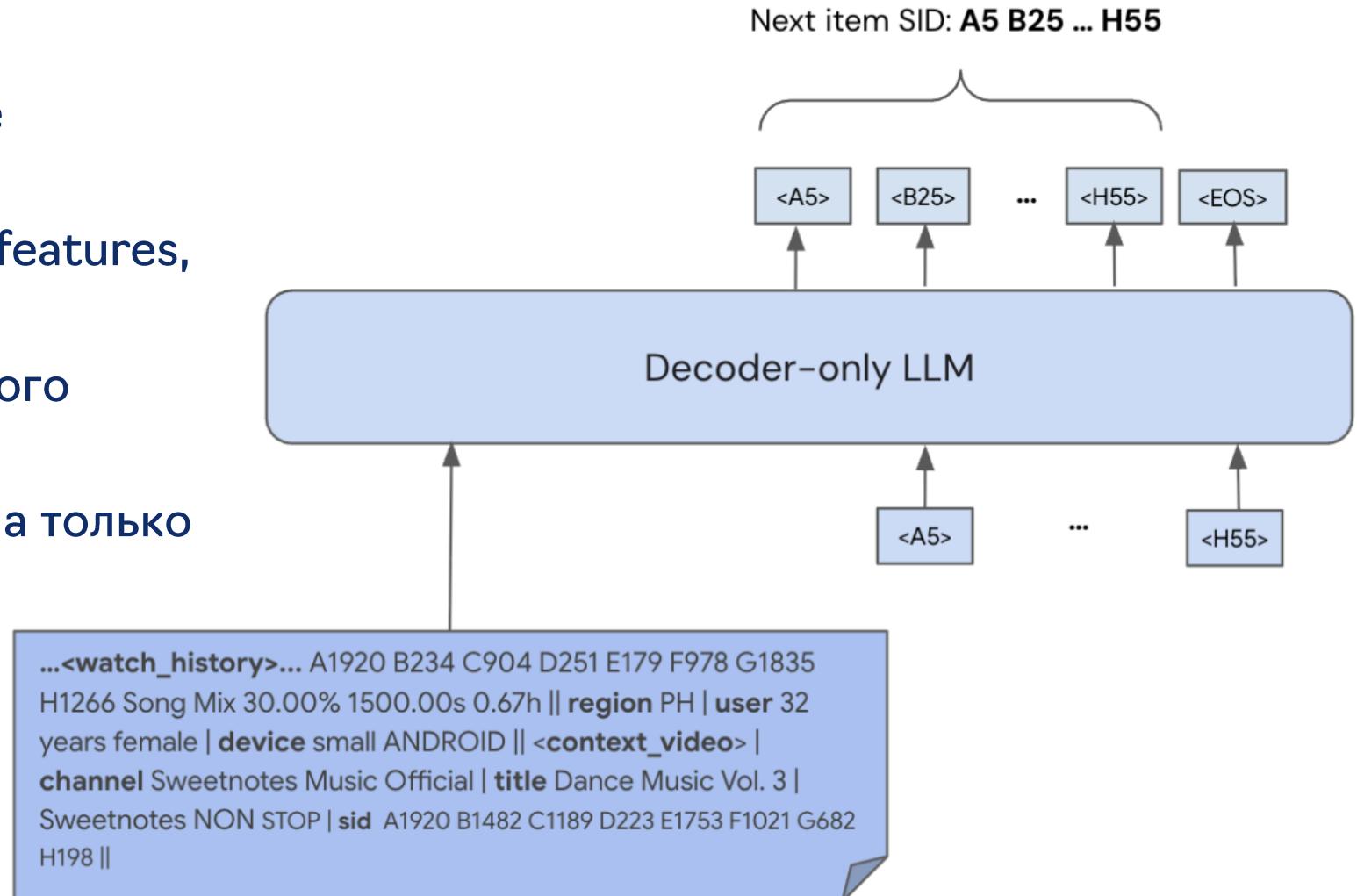
### Randomly-Initialized CPT Output

- - - 100% -1999-11-16

# PLUM: Supervised Fine-tuning

## Дообучение под Generative Retrieval:

- **вход:** [watch history, user features, context features]
- **выход:** Semantic ID целевого айтема
- **учатся не на всех данных, а ТОЛЬКО** на “хороших” кликах
- **внедрили в генерацию кандидатов**



# Что я не успел рассказать

- Графовые нейросети для рекомендаций (e.g., PinSage, TwHIN), гетерогенность
- Foundation модели пользователей на всю экосистему (Netflix, Pinterest, etc)
- HSTU и end-to-end замена счетчиков нейросетями
- Off-policy estimation (and learning)
- RL bandits for exploration
- Re-ranking, slate recommendations (e.g., diffusion models)
- OneRec-Think и LLM-ризонинг для рекомендаций

# Итоги лекции

- Рекомендации – это про масштаб, тяжелые хвосты, холодный старт, distribution drift
- Нейросети используются на всех стадиях:
  - Генерация кандидатов (двухбашенные модели, next item prediction, генеративные модели)
  - Ранжирование (подробней на семинаре)
- Область очень живая и бурно развивается!
  - Генеративные модели, RL, разговорные сценарии, etc
  - Очень много разнообразных моделей, задач

Приходите слушать наш курс! :)

# На семинаре

- Подробный разбор архитектуры нейросетевого ранжирования
- Live демонстрация победы нейросети над градиентным бустингом :)

Подробнее про:

