



RecSys Trends

ШАД, курс по рекомендательным системам, весна 2025

Кирилл Хрыльченко

Структура лекции

1 | Введение

7 | Генеративные модели

2 | Масштабирование

3 | Actions Speak Louder Than Words

4 | LLM

5 | Семантические айдишники

6 | Foundation модели, кроссплатформенность

1

>

Введение

Industry vs Academy

Разные **возможности**. В индустрии:

- › Гораздо больше железа — GPU, CPU, кластеры для распределенного обучения
- › Разительно больше данных (типичный академический датасет — сотни тысяч / десятки миллионов взаимодействий, у нас — миллиарды / сотни миллиардов)
- › Более строгий критерий успеха — влияние на реальный мир (A/B тесты); успешное внедрение. А в академии — успешная публикация
- › Но нет peer review и фундаментальности происходящего

Будем обсуждать больше **индустриальные** тренды.

Откуда брать тренды

Источники информации:

- › Конференции — смотреть proceedings, воркшопы, посещать сами конференции
- › Инженерные блоги
- › Читать тг-каналы и substack'и
- › Просматривать каждый день / неделю / месяц IR секцию arxiv'a

Название, авторы (их аффилиации) и abstract позволяют отсесть много совсем не релевантных статей.

Если проекаете конкретную тему — помогает граф цитирований.

Lagging Behind

Ключевые тренды связаны с нейросетевыми моделями:

- › В плане DL отстаём от других областей
- › Очень много заимствуем из NLP
- › За RL и CV тоже полезно следить
- › Генеративность, семантические айдишники, etc —
появилось не в RecSys

И не забываем про **bitter lesson!**

Тренды

Авторский список:

- › Масштабирование
- › Генеративные модели
- › LLM
- › Семантические айдишники
- › Foundation модели, кросс-платформенность
- › GNN и RL
- › Переосмысление стадии отбора кандидатов (GPU retrieval, mixture of logits, одностадийные рексистемы)

2

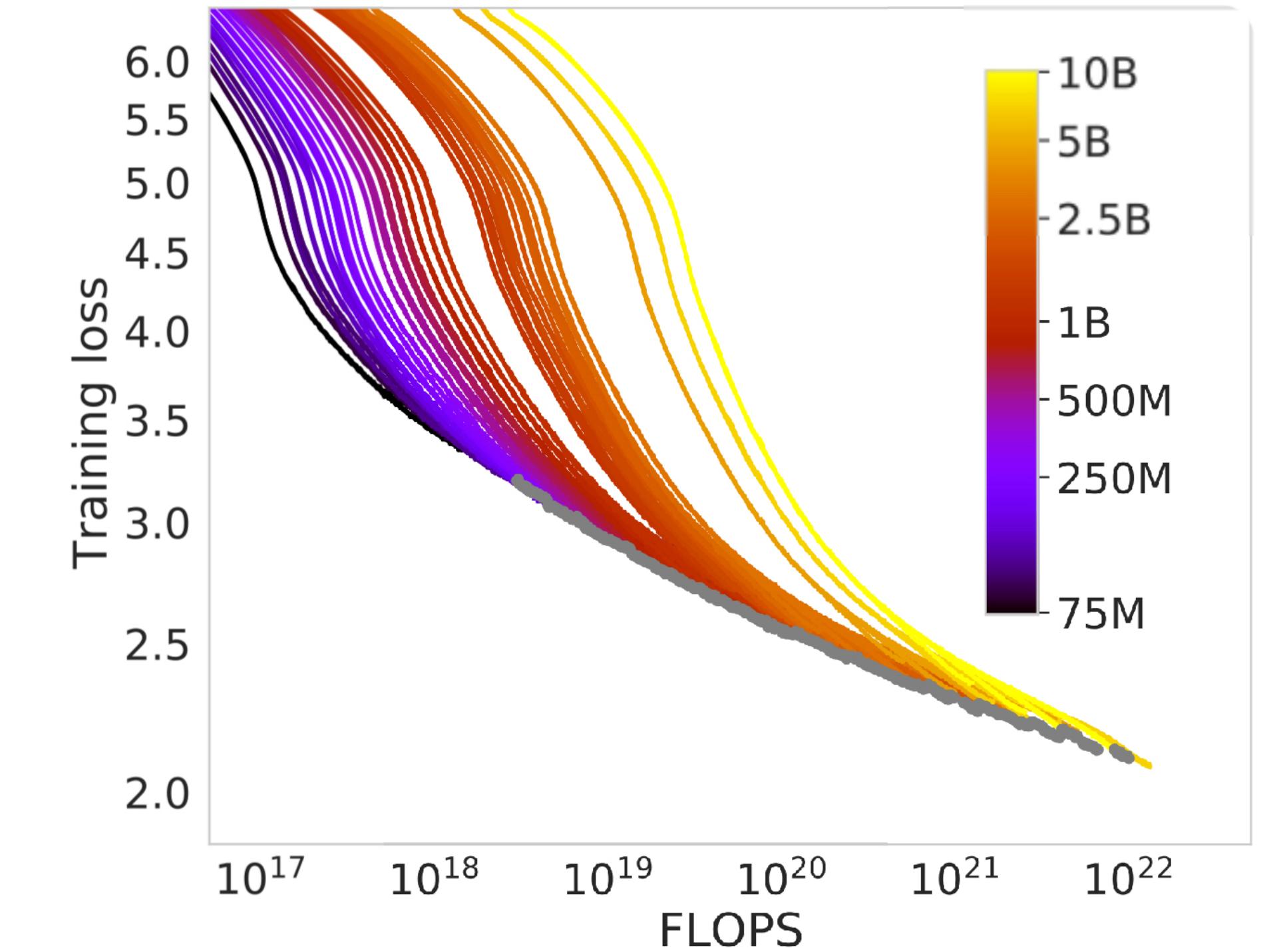
>

Масштабирование

Scaling hypothesis

Чем больше размеры моделей и количество данных, тем выше качество:

- › Последние 10 лет DL-модели постоянно растут по числу параметров и объему обучающих данных
- › Для языковых моделей есть ярко выраженный **scaling law** — предсказуемый рост качества при масштабировании



Training Compute-Optimal Large Language Models

Масштабирование рексистем

Четыре основных оси масштабирования:

- › Размер эмбеддингов
- › Длина контекста
- › Размер датасетов
- › Архитектура энкодеров

Матрицы эмбеддингов

- › Категориальные признаки с кардинальностью от 2 до **миллиардов** (e.g., item id)¹
- › Увеличение размерности эмбеддинга приводит к очень быстрому увеличению размера матрицы эмбеддингов
- › Размерность эмбеддинга - информационный боттлнек¹

Примеры:

- › YouTubeDNN: ~1B параметров²
- › Запрещенная в РФ организация: от 675B³ до 13T⁴ параметров
- › Pinterest: в последних работах⁵ перешел к внедрению больших матриц эмбеддингов

1 [Unified Embedding: Battle-Tested Feature Representations for Web-Scale ML Systems](#)

2 [Deep Neural Networks for YouTube Recommendations](#)

3 [Wukong: Towards a Scaling Law for Large-Scale Recommendation](#)

4 [Software-Hardware Co-design for Fast and Scalable Training of Deep Learning Recommendation Models](#)

5 [Taming the One-Epoch Phenomenon in Online Recommendation System by Two-stage Contrastive ID Pre-training](#)

Датасеты

- › Рекомендации генерируют трафик, сравнимый с сотнями GPT-3 датасетов в день

Статья	Размер датасета
<u>Deep Interest Network for Click-Through Rate Prediction</u>	2B
<u>End-to-end training of Multimodal Model and ranking Model</u>	2.1B
<u>TransAct: Transformer-based Realtime User Action Model for Recommendation at Pinterest</u>	3B
<u>Software-hardware co-design for fast and scalable training of deep learning recommendation models</u>	60B
<u>Deep Neural Networks for YouTube Recommendations</u>	100B
<u>DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems</u>	100B
<u>Wukong: Towards a Scaling Law for Large-Scale Recommendation</u>	146B
<u>Wide & Deep Learning for Recommender Systems</u>	500B

Длина контекста

- › **Число признаков в современных системах колеблется от сотен^{1, 2, 3} до тысяч⁴**
- › **Длина истории пользователя короткая:** в основном около 100^{5, 6, 7, 8} или 256^{9, 10} взаимодействий

1 [Deep Multifaceted Transformers for Multi-objective Ranking in Large-Scale E-commerce Recommender Systems](#)

2 [Wukong: Towards a Scaling Law for Large-Scale Recommendation](#)

3 [Scaling User Modeling: Large-scale Online User Representations for Ads Personalization in ***](#)

4 [Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations](#)

5 [LiRank: Industrial Large Scale Ranking Models at LinkedIn](#)

6 [Deep Neural Networks for YouTube Recommendations](#)

7 [NxtPost: User To Post Recommendations In *** Groups](#)

8 [TransAct: Transformer-based Realtime User Action Model for Recommendation at Pinterest](#)

9 [KuaiFormer: Transformer-Based Retrieval at Kuaishou](#)

10 [PinnerFormer: Sequence Modeling for User Representation at Pinterest](#)

Энкодеры

- › В моделях с ранним связыванием Google от 1М¹ до 68М² параметров
- › Для последовательных рекомендаций используется до 2 трансформерных слоев^{3, 4, 5}, редко 4-5⁶
- › Hidden size порядков сотен^{3, 5}
- › Это трансформеры с <2М параметров

1 [Recommending what video to watch next: a multitask ranking system](#)

2 [Improving Training Stability for Multitask Ranking Models in Recommender Systems](#)

3 [PinnerFormer: Sequence Modeling for User Representation at Pinterest](#)

4 [NxtPost: User To Post Recommendations In *** Groups](#)

5 [TransAct: Transformer-based Realtime User Action Model for Recommendation at Pinterest](#)

6 [KuaiFormer: Transformer-Based Retrieval at Kuaishou](#)

3

>

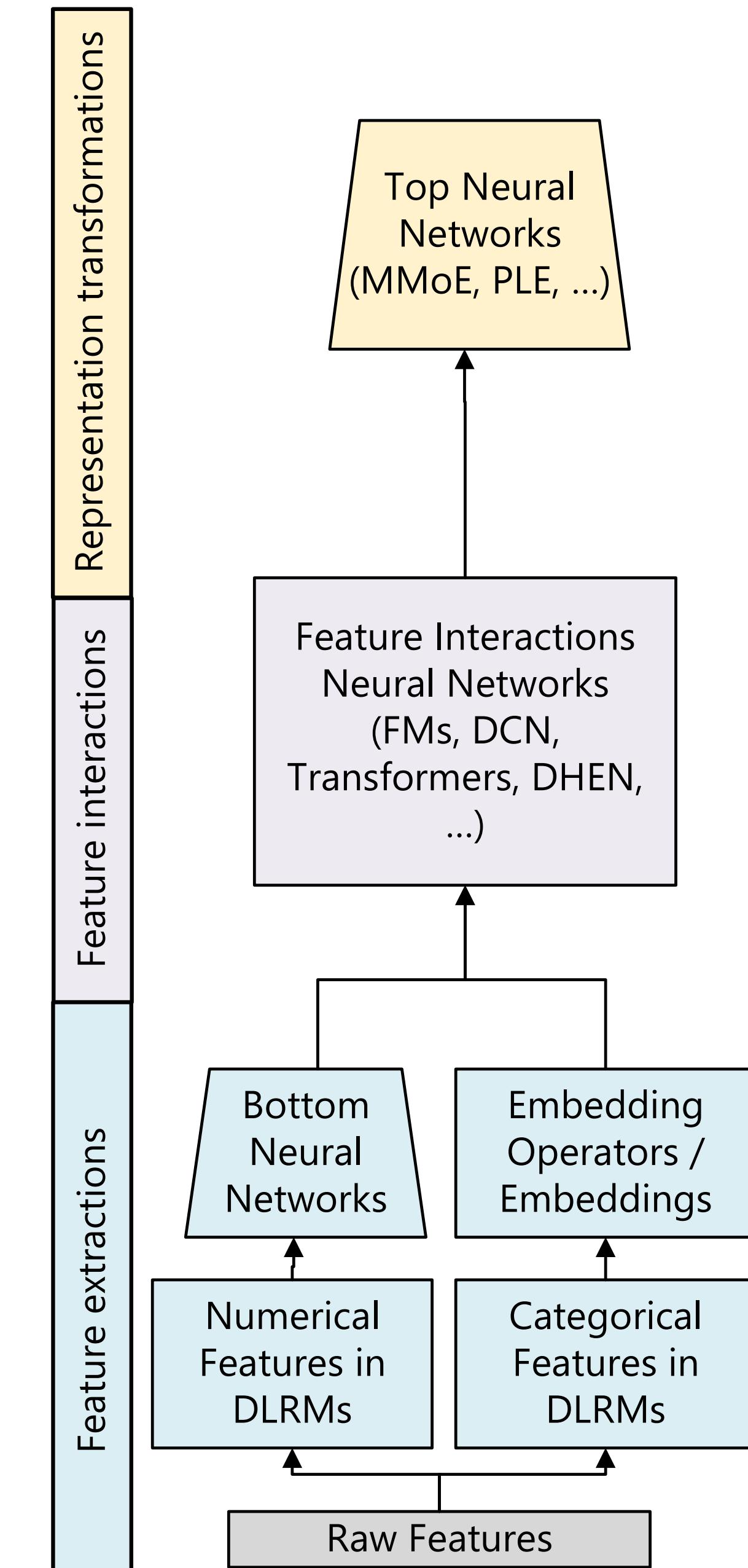
Actions Speak Louder Than Words

Actions Speak Louder than Words: Trillion-
Parameter Sequential Transducers for Generative
Recommendations

DLRM

- › Чем меньше **inductive bias**, тем больше потенциала для скейлинга¹
- › В DLRM много **inductive bias'a** (feature engineering, архитектура)
- › impression-level обучение

... попробуем уменьшить **inductive bias!**



1 Лекция из курса Stanford CS25

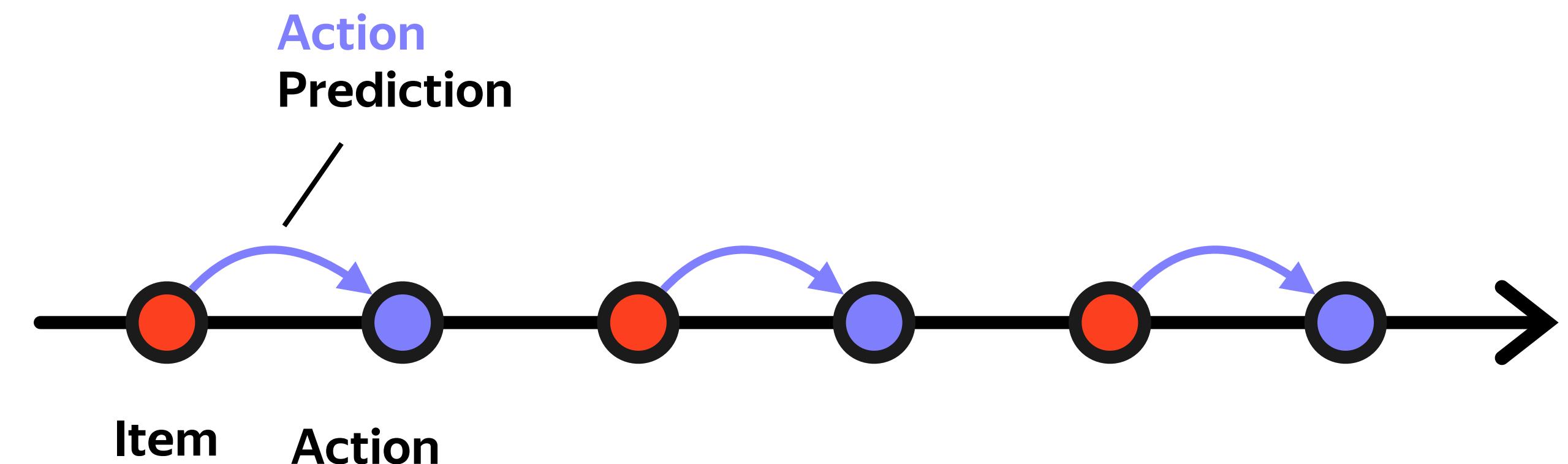
Генеративная постановка

- › Представим пользователя в виде последовательности
- › Добавим статические признаки (возраст, пол, страна, город, etc)
- › Откажемся от вещественных признаков (избавляемся от feature engineering)
- › Item-action interleaving (отдельный токен для айтема, отдельный для экшна)

$x_j \ i_0, a_0, i_1, a_1, \dots, i_{n-1}, a_{n-1}$

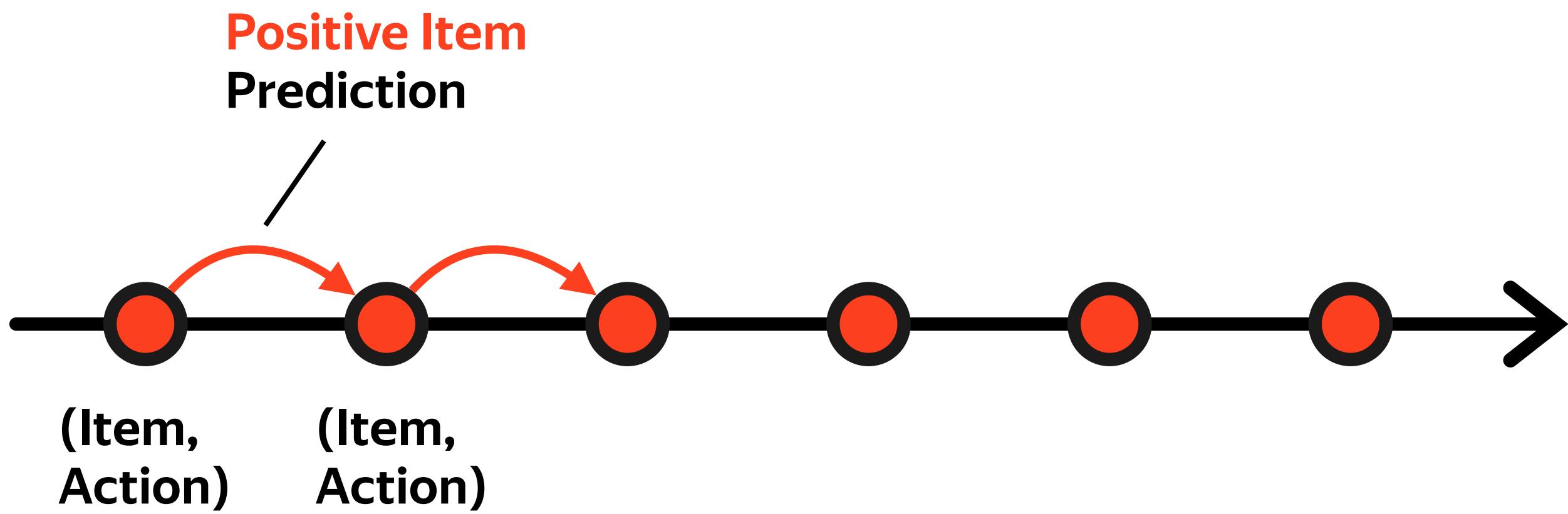
$y_j \ a_0, \emptyset, a_1, \emptyset, \dots, a_{n-1}, \emptyset$

- › Получаем target-aware модель
- › Сильно ускоряемся

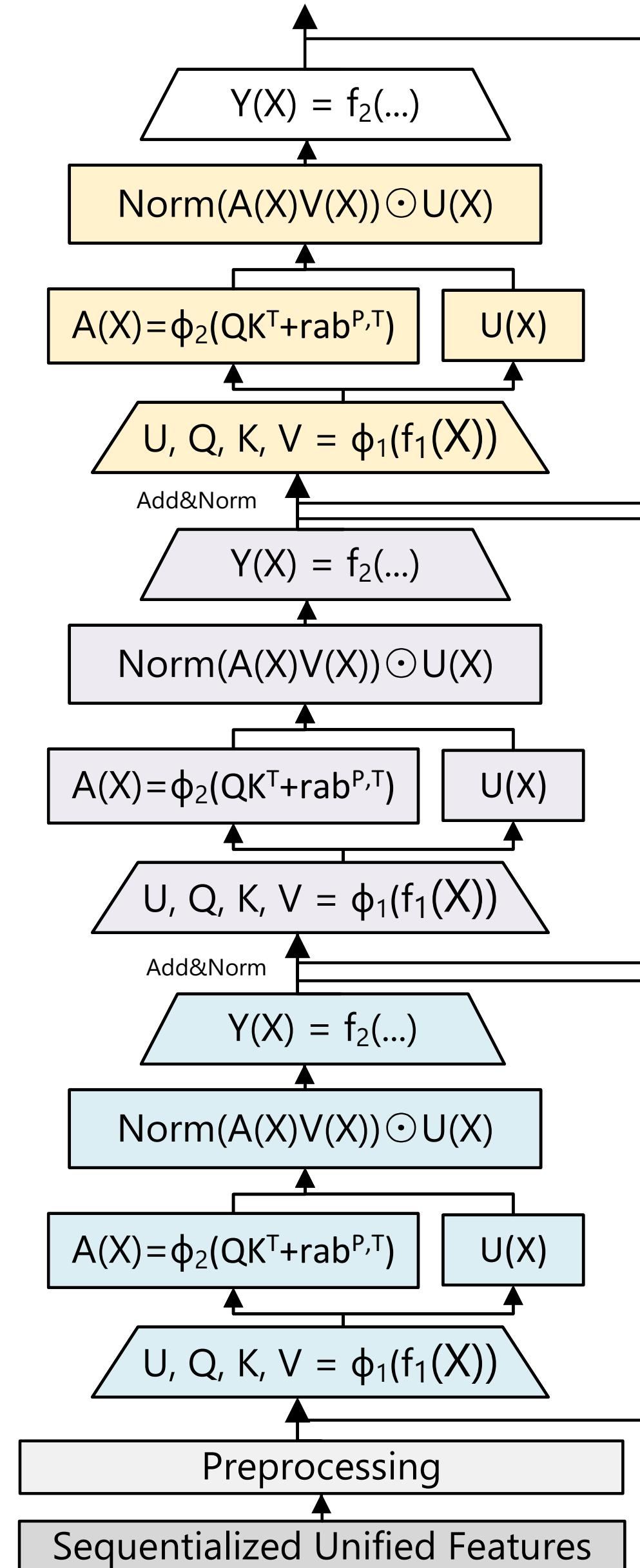


Генеративная постановка

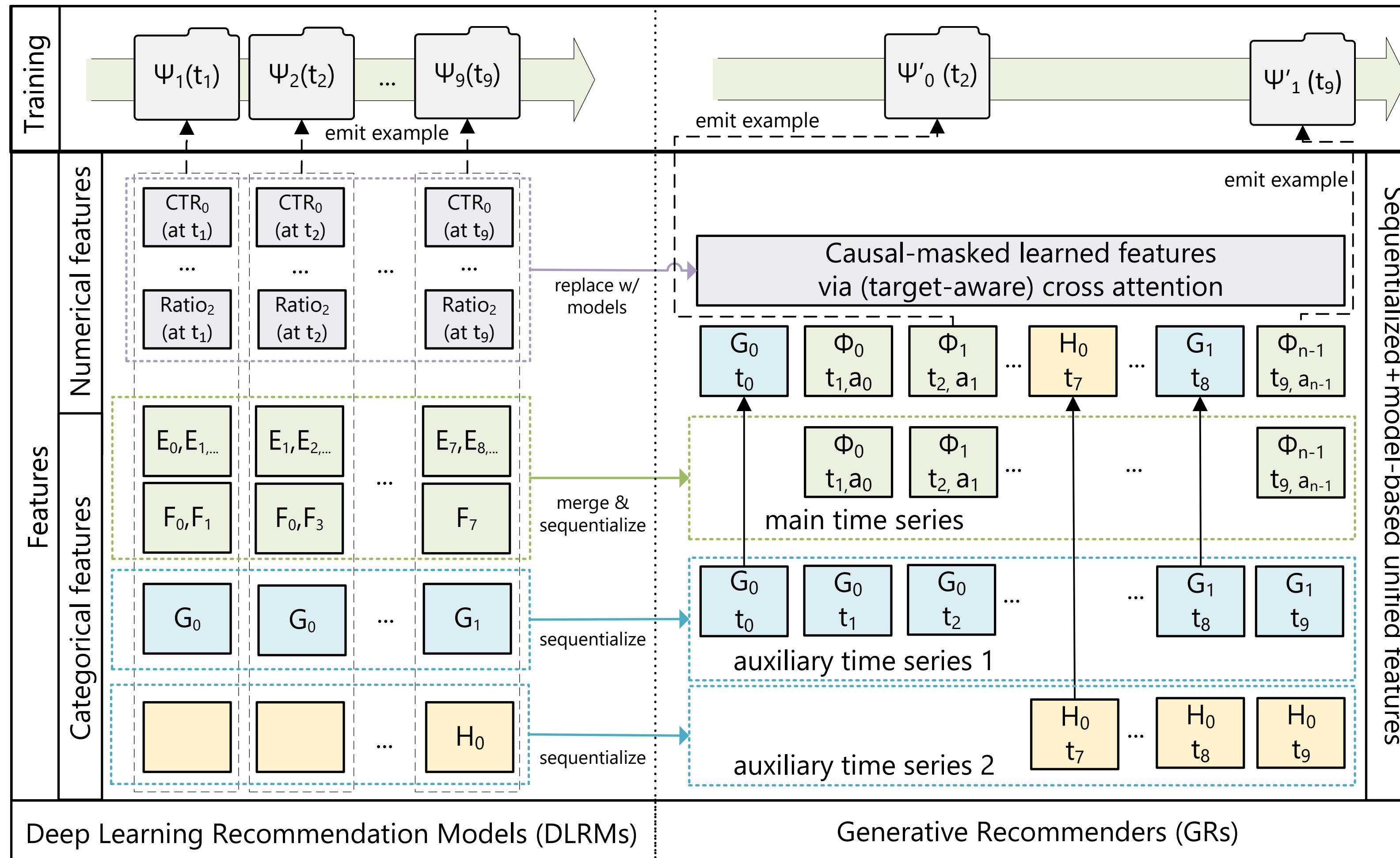
- › Для генерации кандидатов делают **отдельную модель** с другим форматом последовательности
- › Вместо (item, action) будет один вектор $f(\text{item}, \text{action})$
- › Предсказываем только следующие **положительные** взаимодействия



- › Самые полезные признаки в ранжировании — всевозможные **счётчики**, особенно user-item
- › **Софтмакс** плохо улавливает интенсивность элементов в последовательности — **избавимся!**
- › Уберём FFN (сильно уменьшим **память на активации**)
- › Добавить в Attention матрицу U для поэлементных умножений, похожих на всякие **feature interaction** слои



Генеративная постановка



Результаты

Table 4. Evaluations of methods on public datasets in multi-pass, full-shuffle settings.

	Method	HR@10	HR@50	HR@200	NDCG@10	NDCG@200
ML-1M	SASRec (2023)	.2853	.5474	.7528	.1603	.2498
	HSTU	.3097 (+8.6%)	.5754 (+5.1%)	.7716 (+2.5%)	.1720 (+7.3%)	.2606 (+4.3%)
	HSTU-large	.3294 (+15.5%)	.5935 (+8.4%)	.7839 (+4.1%)	.1893 (+18.1%)	.2771 (+10.9%)
ML-20M	SASRec (2023)	.2906	.5499	.7655	.1621	.2521
	HSTU	.3252 (+11.9%)	.5885 (+7.0%)	.7943 (+3.8%)	.1878 (+15.9%)	.2774 (+10.0%)
	HSTU-large	.3567 (+22.8%)	.6149 (+11.8%)	.8076 (+5.5%)	.2106 (+30.0%)	.2971 (+17.9%)
Books	SASRec (2023)	.0292	.0729	.1400	.0156	.0350
	HSTU	.0404 (+38.4%)	.0943 (+29.5%)	.1710 (+22.1%)	.0219 (+40.6%)	.0450 (+28.6%)
	HSTU-large	.0469 (+60.6%)	.1066 (+46.2%)	.1876 (+33.9%)	.0257 (+65.8%)	.0508 (+45.1%)

Результаты

Table 6. Offline/Online Comparison of Retrieval Models.

Methods	Offline HR@K		Online metrics	
	K=100	K=500	E-Task	C-Task
DLRM	29.0%	55.5%	+0%	+0%
DLRM (abl. features)	28.3%	54.3%	—	—
GR (content-based)	11.6%	18.8%	—	—
GR (interactions only)	35.6%	61.7%	—	—
GR (new source)	36.9%	62.4%	+6.2%	+5.0%
GR (replace source)			+5.1%	+1.9%

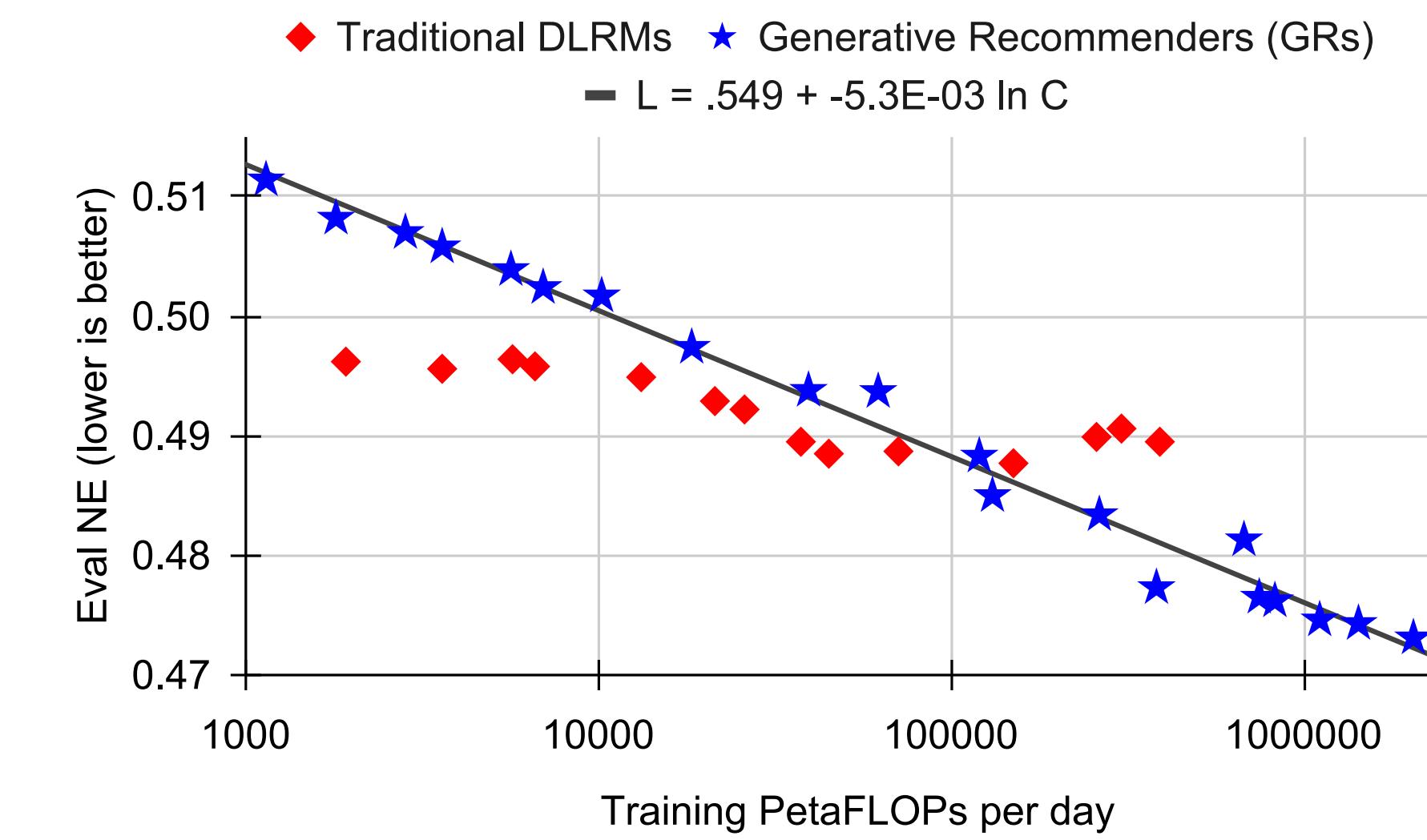
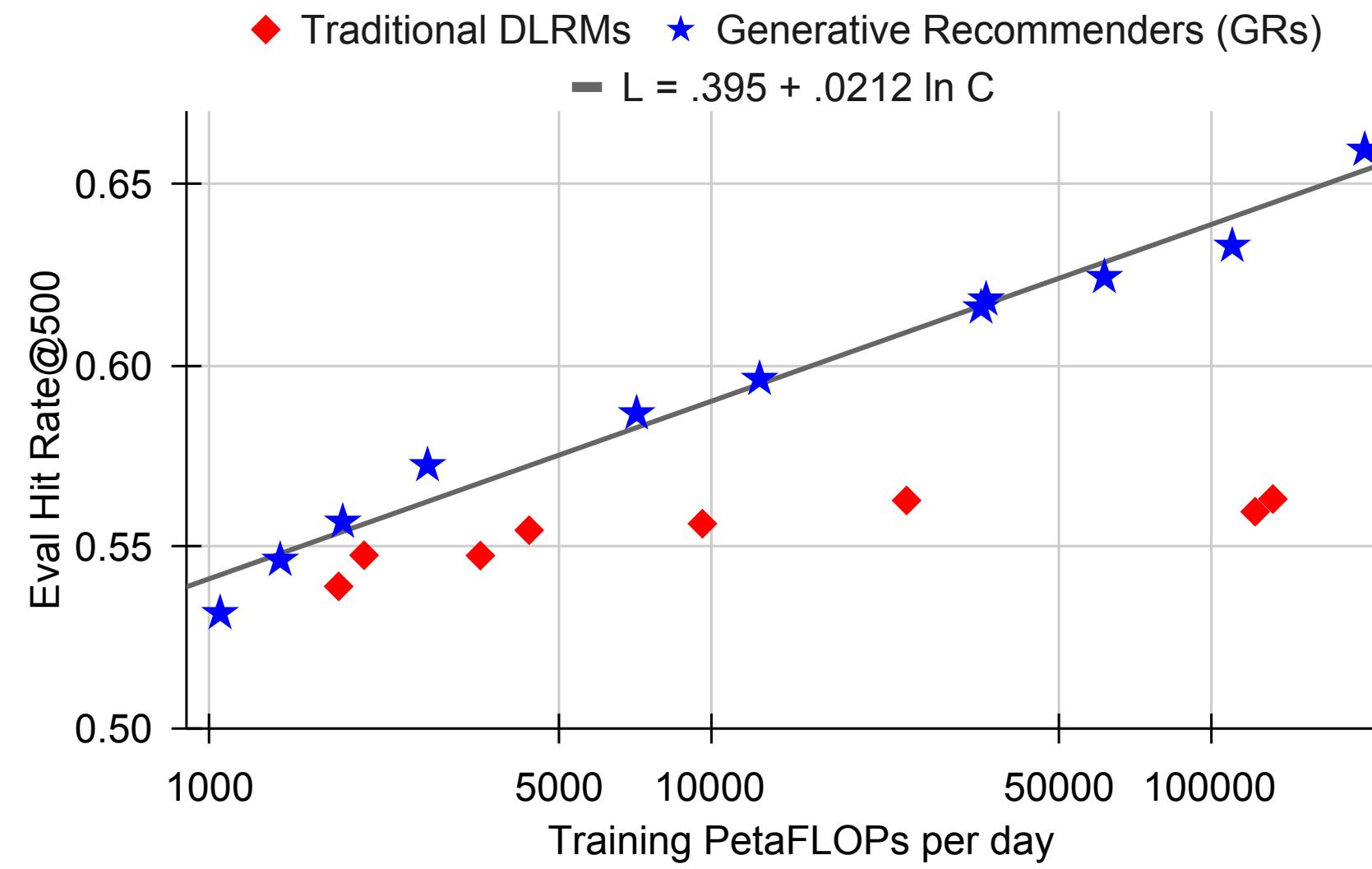
Table 7. Offline/Online Comparison of Ranking Models.

Methods	Offline NEs		Online metrics	
	E-Task	C-Task	E-Task	C-Task
DLRM	.4982	.7842	+0%	+0%
DLRM (DIN+DCN)	.5053	.7899	—	—
DLRM (abl. features)	.5053	.7925	—	—
GR (interactions only)	.4851	.7903	—	—
GR	.4845	.7645	+12.4%	+4.4%

Table 5. Evaluation of HSTU, ablated HSTU, and Transformers on industrial-scale datasets in one-pass streaming settings.

Architecture	Retrieval log pplx.	Ranking (NE)	
		E-Task	C-Task
Transformers	4.069	NaN	NaN
HSTU (-rab ^{p,t} , Softmax)	4.024	.5067	.7931
HSTU (-rab ^{p,t})	4.021	.4980	.7860
Transformer++	4.015	.4945	.7822
HSTU (original rab)	4.029	.4941	.7817
HSTU	3.978	.4937	.7805

Результаты



Проблемы HSTU

- › Все еще **небольшой скейлинг** по энкодеру, максимальная конфигурация - 176М параметров
- › **Не удалось воспроизвести** профит от HSTU архитектуры в индустрии
- › Все еще **две отдельных постановки retrieval / ranking**, не полная генеративность
- › Довольно **тяжелый сценарий внедрения** ранжирования (target-aware)

4

>

LLM

LLM x RecSys

Мотивация:

- › Альтернативный способ масштабирования
- › Решаем проблему **холодного старта**
- › Используем **внешние знания о мире**
- › **Объяснения** рекомендаций
- › **Project Astra**¹, диалог как интерфейс

1 <https://deepmind.google/models/project-astra/>

Естественная эволюция

- › Текстовый “промпт” с просьбой что-то порекомендовать (релевантность без персонализации)
- › **Персонализированный** промпт (добавляем историю в текстовом виде)
- › Сворачиваем отдельной, более “классической” коллаборативной моделью историю пользователя в вектор
- › Можно решать задачу ранжирования (сразу подаем список кандидатов)
- › (LLM для кодирования айтемов в векторы на входе в классическую модель)

Проблемы

- › **Open-set кандидатогенерация** — надо как-то заматчить сгенерированное на реальный каталог (дополнительная стадия матчинга / закладывание знания о каталоге в сам трансформер)
- › **Нестабильность¹** (меняешь слово в промпте — меняется порядок ранжирования поданных на вход кандидатов)
- › **Нужно много ресурсов**
- › **Language-recommendation vocabulary gap**

¹ [Large Language Models are Not Stable Recommender Systems](#)

LLM - RecSys Gap

- › Если не файнтюнить LLM на рекомендательный трафик (то есть не привносить коллаборативность) — **плохое качество** рекомендаций
- › Если файнтюнить - **теряем способности естественного языка**
- › Как обучить так, чтобы были и способности естественного языка, и лучшее качество рекомендаций?
- › Deepmind файнтюнит **Gemini** на рекомендации

5

>

Семантические айдишники

Semantic IDs

- › ID-based теряет генерализацию на похожих по контенту айтемах
- › **Hashing trick** и коллизии помогают бороться с переобучением на тяжелом хвосте, но ухудшают качество на холодных айтемах
- › Если использовать **контентные векторы** — падает качество, теряем полезную меморизацию
- › Хотим хороший баланс между меморизацией и генерализацией

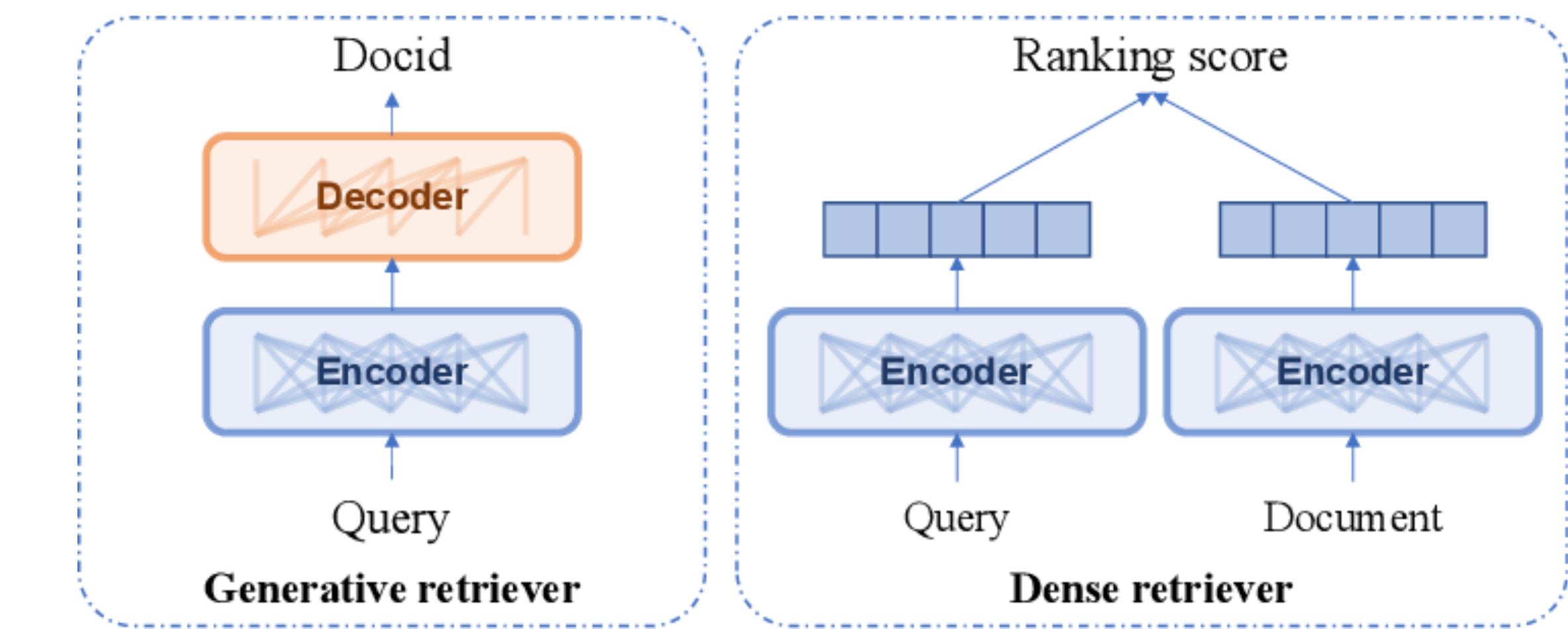
Semantic IDs

- › В Гугле не любят работать с большими матрицами эмбеддингов (см. unified embeddings и vector quantization ресерч)
- › **Проблема полного софтмакса:** сложно учиться на задачу предсказания следующего взаимодействия в истории пользователя для больших каталогов айтемов
- › Боремся с **vocabulary mismatch**: хотим совмещать естественный язык и историю событий в жизни пользователя. **Project Astra!**

Generative Retrieval

Изначально придумали для информационного поиска:

- › Раньше использовали двухбашенные модели: вектор запроса, вектор документа, скалярное произведение
- › Теперь будем глядя на запрос генерировать документ
- › **Подход 1:** генерируем название и метаданные документа
- › **Подход 2:** сопоставляем каждому документу последовательность айдишников, генерируем эту последовательность



Из статьи [ROGER: Ranking-Oriented Generative Retrieval](#)

Вариационные автокодировщики

- › **VAE¹**: хотели научиться генерировать качественные картинки. У похожих картинок будут похожие латентные представления!
- › **VQ-VAE²**: хотим при генерации использовать дискретную последовательность кодов. Это больше похоже на язык
- › **RQ-VAE³**: делаем иерархичную последовательность кодов, чтобы у похожих айтемов совпадали префиксы. Предложили для аудиосигнала!
- › Хорошо комбинируется с мультимодальностью: весь “нетекстовый” сигнал (e.g. картинки, аудио, видео) кодируется в семантические айдишники

1 [An Introduction to Variational Autoencoders](#)

2 [Neural Discrete Representation Learning](#)

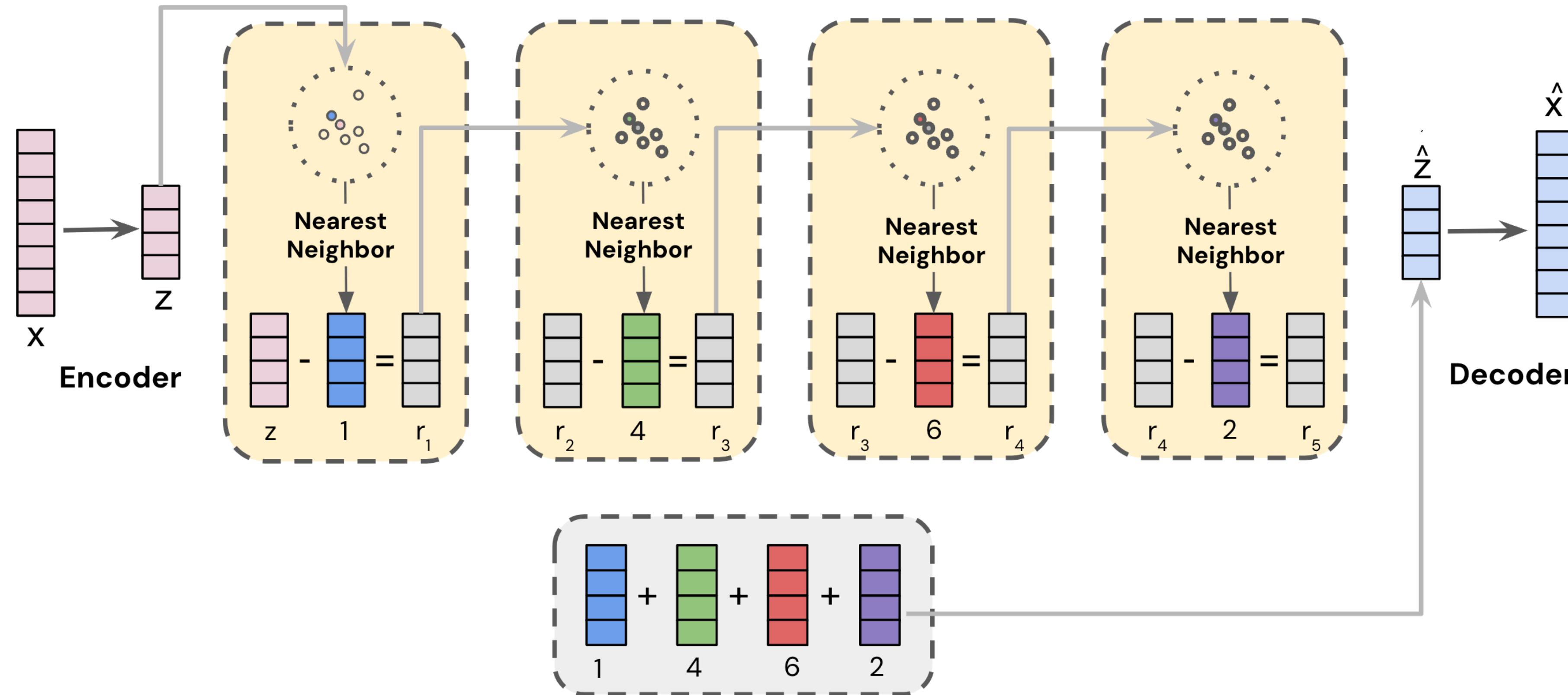
3 [SoundStream: An End-to-End Neural Audio Codec](#)

Semantic IDs

Применим вариационные автокодировщики к векторам айтемов:

- › **Vocabulary mismatch:** айтем — это более высокоуровневые сущности чем токены естественного языка, нужно их тоже “токенизировать” на более базовые концепты, представленные айдишниками низкой кардинальности
- › **Проблемы на холодных и редких айтемах:** у похожих айтемов будут похожие последовательности айдишников. Теперь коллизии — семантические!
- › **Проблема полного софтмакса и с потреблением памяти:** айдишники низкой кардинальности проще моделировать. Autoregressive decoding + beam search вместо ANNS. Теперь индекс — это сама модель!

RQ-VAE



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

RQ-VAE

Как работает кодировщик:

- › Получаем для айтема векторное представление
- › Находим ближайший к нему элемент из первого кодбука по евклидовому расстоянию
- › Вычитаем из исходного вектора вектор этого элемента кодбука (получаем **residual**), затем ищем к нему ближайший элемент из **второго** кодбука
- › Повторяем столько раз, сколько хотим айдишников
- › Новое векторное представление айтема — сумма всех отобранных элементов кодбуков (но необязательно)

RQ-VAE

Детали:

- › При обучении подаем это новое векторное представление (сумму residual'ов) на вход декодеру
- › Лучше использовать для разных residual'ов разные небольшие кодбуки вместо одного большого кодбука, чтобы уменьшить коллизии
- › Итоговый лосс — сумма reconstruction лосса (MSE между векторами) и лосса, сближающего центроиды и соответствующие residual'ы
- › Используем STE чтобы делать бэкпроп по reconstruction лоссу для энкодера

Public Setup

- › Взяли **датасеты Амазона** (средняя длина историй 8)
- › Прогнали (title, price, brand, category) через **Sentence-T5**, получили векторы размерности 768
- › Обучили **RQ-VAE** поверх этих векторов с длиной семантического кодирования равной 3
- › Максимум коллизий айтемов в рамках одной цепочки семантических айдишников — 40 айтемов. Добавили в конец цепочки еще один айдишник со значением от 0 до 40
- › Возможны цепочки семантических айдишников, не соответствующие ни одному реальному айтему из каталога

Public Setup

- › Используют **энкодер-декодер**: энкодер обрабатывает историю, декодер генерирует семантические айдишники следующего айтема
- › Добавили **обучаемый эмбеддинг пользователя** по хэшу от айдишника кардинальности 2000 O_o
- › Не заводили бейзлайны сами (кроме P5), взяли для них готовые метрики из других статей
- › Получили адекватное качество (но в реальности непонятно насколько хорошее)

Результаты

Table 1: Performance comparison on sequential recommendation. The last row depicts % improvement with TIGER relative to the best baseline. Bold (underline) are used to denote the best (second-best) metric.

Methods	Sports and Outdoors				Beauty				Toys and Games			
	Recall @5	NDCG @5	Recall @10	NDCG @10	Recall @5	NDCG @5	Recall @10	NDCG @10	Recall @5	NDCG @5	Recall @10	NDCG @10
P5 [8]	0.0061	0.0041	0.0095	0.0052	0.0163	0.0107	0.0254	0.0136	0.0070	0.0050	0.0121	0.0066
Caser [33]	0.0116	0.0072	0.0194	0.0097	0.0205	0.0131	0.0347	0.0176	0.0166	0.0107	0.0270	0.0141
HGN [25]	0.0189	0.0120	0.0313	0.0159	0.0325	0.0206	0.0512	0.0266	0.0321	0.0221	0.0497	0.0277
GRU4Rec [11]	0.0129	0.0086	0.0204	0.0110	0.0164	0.0099	0.0283	0.0137	0.0097	0.0059	0.0176	0.0084
BERT4Rec [32]	0.0115	0.0075	0.0191	0.0099	0.0203	0.0124	0.0347	0.0170	0.0116	0.0071	0.0203	0.0099
FDSA [42]	0.0182	0.0122	0.0288	0.0156	0.0267	0.0163	0.0407	0.0208	0.0228	0.0140	0.0381	0.0189
SASRec [17]	0.0233	0.0154	0.0350	0.0192	0.0387	<u>0.0249</u>	0.0605	0.0318	<u>0.0463</u>	<u>0.0306</u>	0.0675	0.0374
S ³ -Rec [44]	<u>0.0251</u>	<u>0.0161</u>	<u>0.0385</u>	<u>0.0204</u>	<u>0.0387</u>	0.0244	<u>0.0647</u>	<u>0.0327</u>	0.0443	0.0294	<u>0.0700</u>	<u>0.0376</u>
TIGER [Ours]	0.0264	0.0181	0.0400	0.0225	0.0454	0.0321	0.0648	0.0384	0.0521	0.0371	0.0712	0.0432
	+5.22%	+12.55%	+3.90%	+10.29%	+17.31%	+29.04%	+0.15%	+17.43%	+12.53%	+21.24%	+1.71%	+14.97%

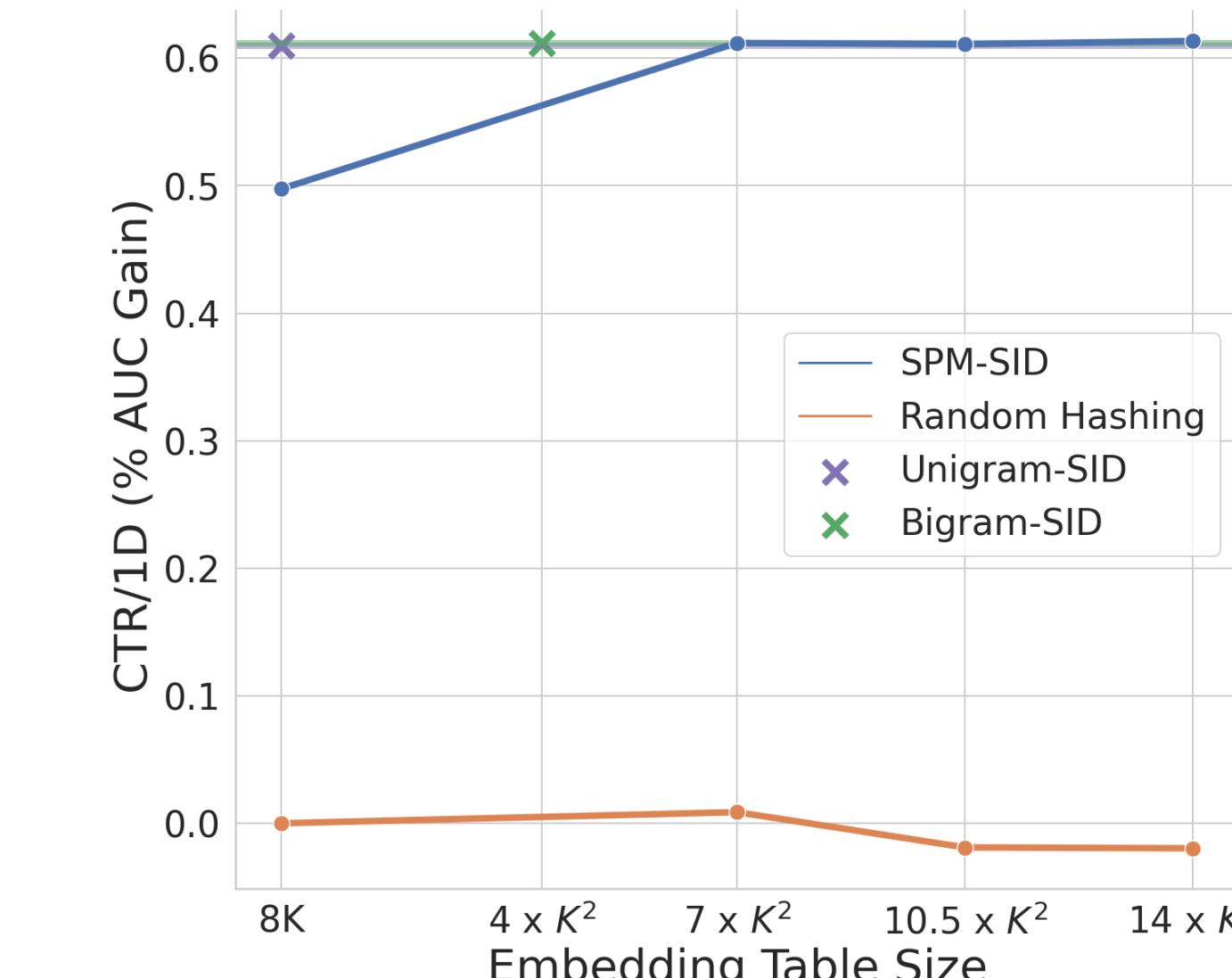
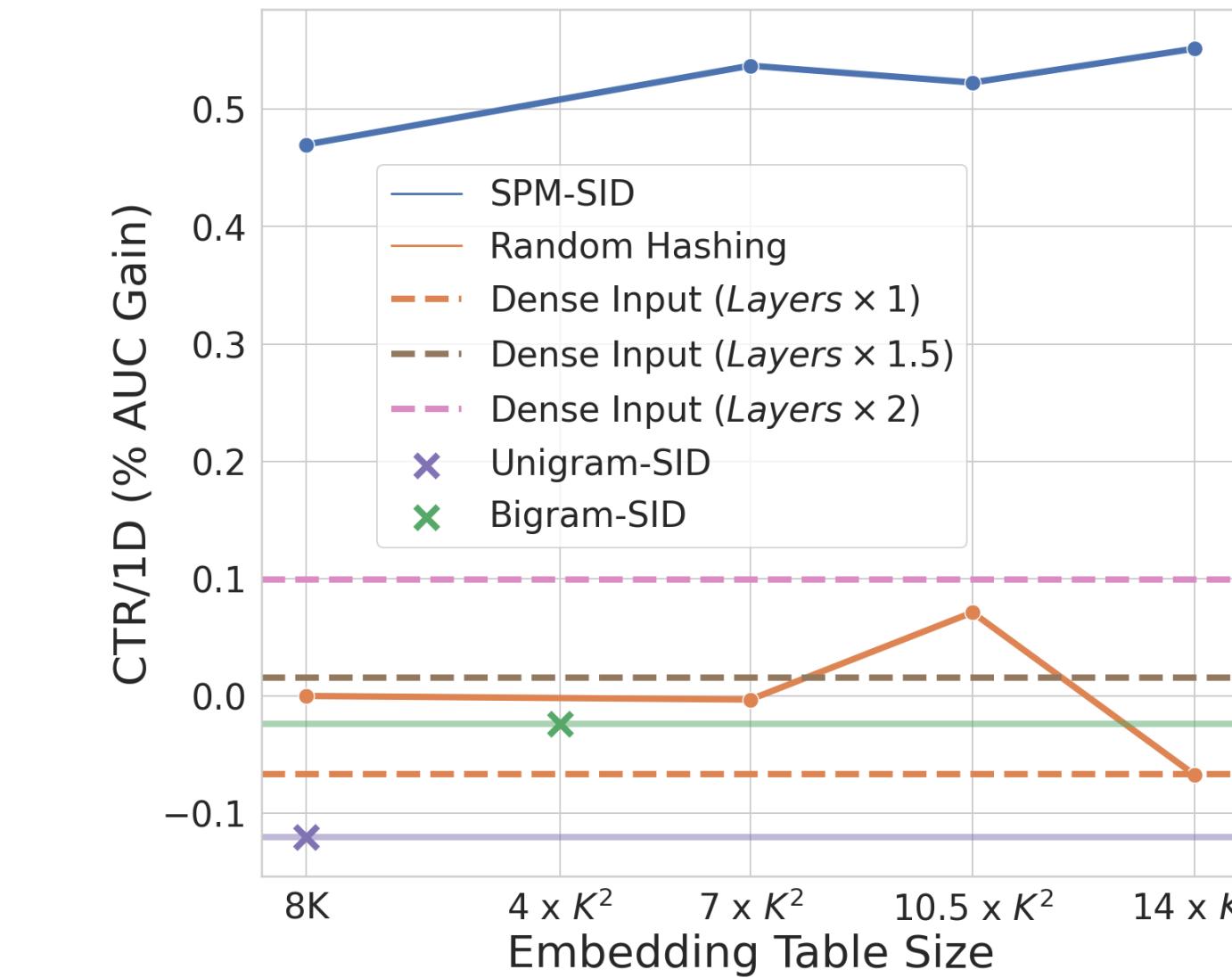
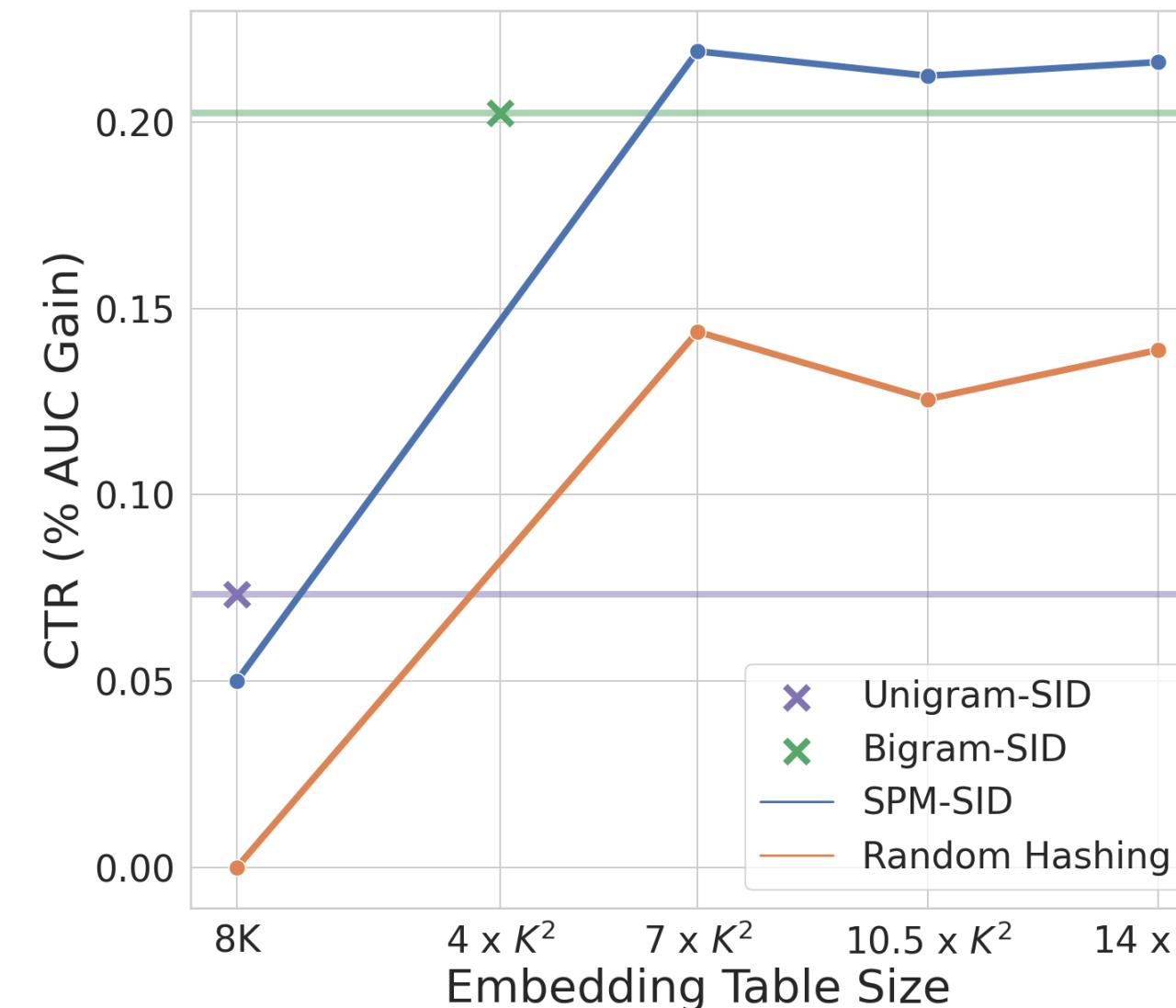
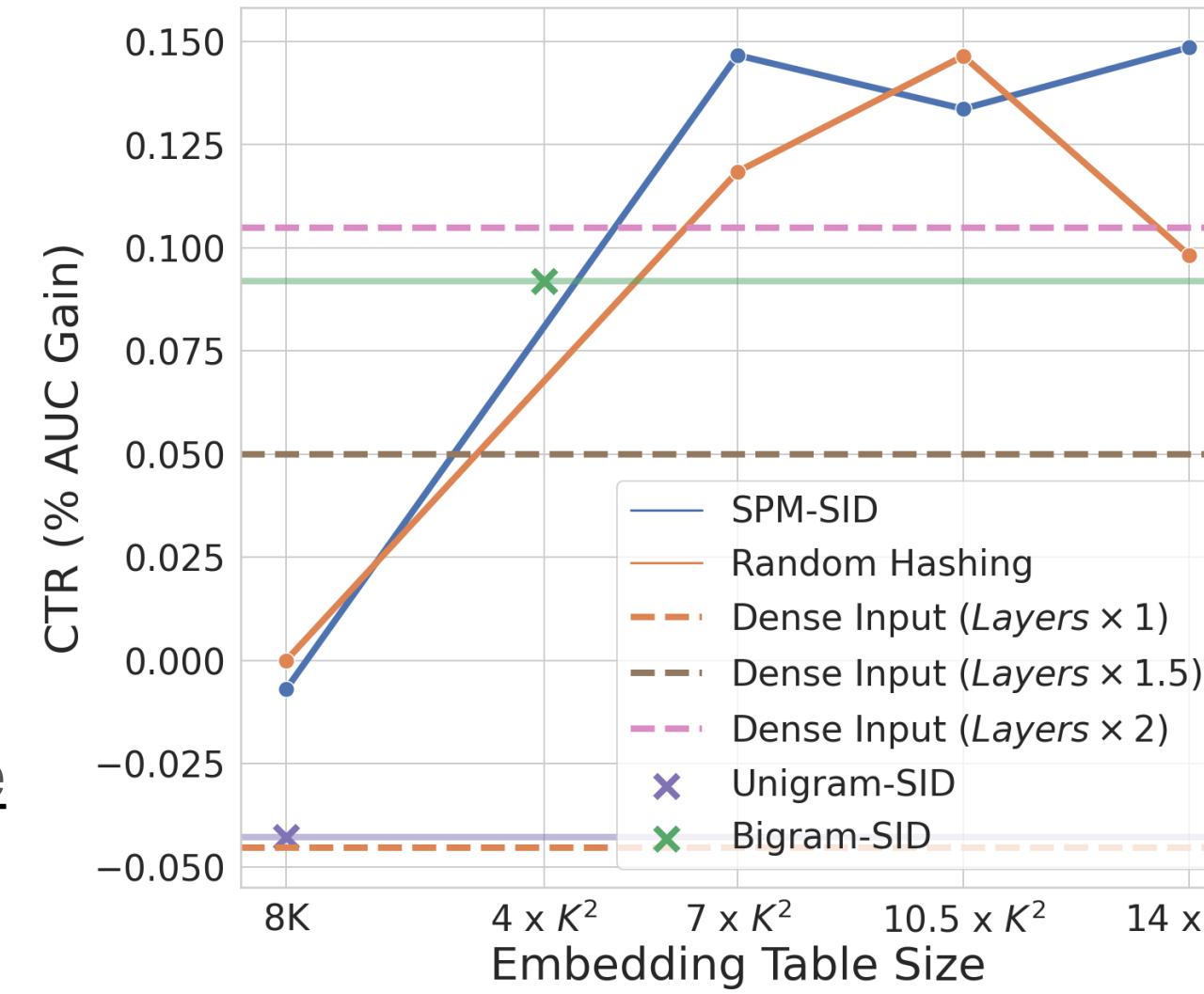
Из статьи [Recommender Systems with Generative Retrieval](#)

Production Setup

- › Есть большой корпус айтемов (e.g., **миллиарды** видеозаписей в YouTube)
- › Есть **хороший контентный энкодер** для айтемов (VideoBERT), с помощью которого умеем получать очень широкие (e.g., dim 2048) эмбеды айтемов
- › Обучают на этих векторах RQ-VAE (не учитываем популярности айтемов); длина кодирования — **8 айдишников** (где-то даже 16)
- › Используют семантические айдишники на входе в ранжирующую модель; применяют к ним sentencepiece
- › Также тюнят Gemini на генерацию таких айдишников

Результаты

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



Проблемы

- › **Distribution drift** для семантического энкодера; плохо что RQ-VAE зафрижен и не дообучается
- › Нужен **хороший контентный энкодер**; это сама по себе сложная задача
- › **Hourglass phenomenon¹** — RQ-VAE не очень стабильный; промежуточные семантические айдишники могут “схлопнуться” в небольшое количество используемых значений

6

>

Foundation модели, кроссплатформенность

Foundation Model x Кроссплатформенность

Мотивация:

- › Легче развивать и поддерживать одну модель
- › Knowledge Transfer

Тренды:

- › Объединение поиска и рекомендаций¹
- › Объединение всех данных экосистемы, даунстрим файнчуны (e.g. Netflix²)

1 [Bridging Search and Recommendation in Generative Retrieval: Does One Task Help the Other?](#)

2 <https://netflixtechblog.com/foundation-model-for-personalized-recommendation-1a0bd8e02d39>

7

>

Генеративные модели

OneRec

Обучают модель, **генерирующую следующую сессию** пользователя:

- › **Энкодер-декодер**: в энкодер подаём историю, генерируем следующую сессию
- › Учимся генерировать **только хорошие сессии**, подпадающие под определенные критерии — суммарное время просмотра больше порога, просмотрено не меньше 5 видео, поставили лайки
- › Доступны **хорошие мультимодальные эмбеддинги видеороликов**
- › Вместо RQ-VAE используют обычный **k-means** для получения айдишников
- › В начале цепочки семантических айдишников вставляют **BoS-токен**
- › Учатся на кросс-энтропию (полный софтмакс)

OneRec

Делают RL:

- › Обучают **sessionwise reward** модель для оценки качества сессий (всякие sessionwise метрики типа суммарного времени просмотра)
- › Используют DPO, чтобы дотюнить модельку (как RLHF, только вместо человеческих оценок — прогноз reward модели)

Гиперпараметры:

- › Длина семантического кодирования — 3 айдишника
- › Длина истории — 256 событий
- › Генерируют следующие 5 айтемов
- › Внедрили как одностадийную рексистему!

Table 2: The absolute improvement of OneRec compared to the current multi-stage system in the online A/B testing setting.

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%

Table 1: Offline performance of our proposed OneRec (green) with pointwise methods (brown), listwise methods (blue) and preference alignment methods (yellow). Best results are in bold, sub-optimal results are underlined. Metrics with \uparrow indicate higher is better, while \downarrow indicates lower is better.

Model	Watching-Time Metrics				Interaction Metrics			
	swt \uparrow		vtr \uparrow		wtr \uparrow		ltr \uparrow	
	mean	max	mean	max	mean	max	mean	max
<i>Pointwise Discriminative Method</i>								
SASRec	0.0375 \pm 0.002	0.0803 \pm 0.005	0.4313 \pm 0.013	0.5801 \pm 0.013	0.00294 \pm 0.001	<u>0.00978\pm0.001</u>	0.0314 \pm 0.002	0.0604 \pm 0.004
BERT4Rec	0.0336 \pm 0.002	0.0706 \pm 0.004	0.4192 \pm 0.014	0.5633 \pm 0.013	0.00281 \pm 0.001	<u>0.00932\pm0.001</u>	0.0316 \pm 0.002	0.0606 \pm 0.004
FDSA	0.0325 \pm 0.002	0.0683 \pm 0.005	0.4145 \pm 0.015	0.5588 \pm 0.014	0.00271 \pm 0.001	<u>0.00921\pm0.001</u>	0.0313 \pm 0.002	0.0604 \pm 0.003
<i>Pointwise Generative Method</i>								
TIGER-0.1B	0.0879 \pm 0.007	0.1286 \pm 0.010	0.5826 \pm 0.016	0.6625 \pm 0.017	0.00277 \pm 0.001	<u>0.00671\pm0.001</u>	0.0316 \pm 0.004	0.0541 \pm 0.007
TIGER-1B	0.0873 \pm 0.006	0.1368 \pm 0.010	0.5827 \pm 0.015	0.6776 \pm 0.015	0.00292 \pm 0.001	<u>0.00758\pm0.001</u>	0.0323 \pm 0.004	0.0579 \pm 0.008
<i>Listwise Generative Method</i>								
OneRec-0.1B	0.0973 \pm 0.010	0.1501 \pm 0.015	0.6001 \pm 0.021	0.6981 \pm 0.021	0.00326 \pm 0.001	<u>0.00870\pm0.001</u>	0.0349 \pm 0.009	0.0642 \pm 0.015
OneRec-1B	0.0991 \pm 0.008	0.1529 \pm 0.012	0.6039 \pm 0.020	0.7013 \pm 0.020	<u>0.00349\pm0.001</u>	<u>0.00919\pm0.002</u>	0.0360 \pm 0.005	<u>0.0660\pm0.008</u>
<i>Listwise Generative Method with Preference Alignment</i>								
OneRec-1B+DPO	0.1014 \pm 0.010	<u>0.1595\pm0.015</u>	<u>0.6127\pm0.017</u>	<u>0.7116\pm0.016</u>	0.00339 \pm 0.001	<u>0.00896\pm0.001</u>	0.0351 \pm 0.004	0.0644 \pm 0.008
OneRec-1B+IPO	0.0979 \pm 0.003	0.1528 \pm 0.005	0.6000 \pm 0.007	0.7012 \pm 0.007	0.00335 \pm 0.001	<u>0.00905\pm0.001</u>	0.0350 \pm 0.003	0.0654 \pm 0.004
OneRec-1B+cDPO	0.0993 \pm 0.006	0.1547 \pm 0.008	0.6030 \pm 0.011	0.7030 \pm 0.009	0.00339 \pm 0.001	<u>0.00901\pm0.001</u>	0.0355 \pm 0.006	0.0652 \pm 0.009
OneRec-1B+rDPO	0.1005 \pm 0.006	0.1555 \pm 0.008	0.6071 \pm 0.014	0.7059 \pm 0.011	0.00339 \pm 0.001	<u>0.00899\pm0.001</u>	0.0357 \pm 0.004	0.0657 \pm 0.006
OneRec-1B+CPO	0.0993 \pm 0.008	0.1538 \pm 0.012	0.6045 \pm 0.021	0.7029 \pm 0.018	0.00343 \pm 0.001	<u>0.00911\pm0.002</u>	0.0357 \pm 0.008	0.0659 \pm 0.014
OneRec-1B+simPO	0.0995 \pm 0.008	0.1536 \pm 0.013	0.6047 \pm 0.016	0.7022 \pm 0.015	<u>0.00349\pm0.001</u>	<u>0.00918\pm0.001</u>	0.0360 \pm 0.005	0.0659 \pm 0.008
OneRec-1B+S-DPO	<u>0.1021\pm0.008</u>	0.1575 \pm 0.013	0.6096 \pm 0.016	0.7070 \pm 0.015	0.00345 \pm 0.001	<u>0.00909\pm0.001</u>	<u>0.0361\pm0.004</u>	0.0659 \pm 0.008
OneRec-1B+IPA	0.1025\pm0.009	0.1933\pm0.017	0.6141\pm0.020	0.7646\pm0.021	0.00354\pm0.001	0.00992\pm0.001	0.0397\pm0.004	0.1203\pm0.010

PinRec

Взяли за основу **Pinnerformer++**:

- › Тоже sampled softmax loss с logQ-correction и mixed negative sampling
- › Также кодируют последовательность (кодируют время, поверхность взаимодействия, etc)
- › Добавили к PinSage эмбедам еще **обучаемые эмбеддинги пинов**
- › В истории есть **поисковые запросы** (эмбеддинг из OmniSearchSage)

Свой подход:

- › **Не используют семантические айдишники**, ходят в ANNS индекс за айтемами
- › **Не используют RL** чтобы дотюнить генерацию на нужные свойства
- › **Multi-token prediction**: предсказывают не следующее событие, а выбирают какой-то будущий таймстемп и затем предсказывают события в окне после него
- › **Outcome conditioning**: при обучении обуславливают на тип действия и время, с которого начинается окно; при применении выбирают нужные condition'в

PinRec

Table 1: Comparison of baselines and PinRec variants (unordered recall @ 10) across major surfaces at Pinterest.

Model	Homefeed	Related Pins	Search
TIGER [20]	0.208	0.230	0.090
SASRec [11]	0.382	0.426	0.142
PinnerFormer [16]	0.461	0.412	0.257
PinRec-UC	0.608	0.521	0.350
PinRec-OC	0.625	0.537	0.352
PinRec-{MT, OC}	0.676	0.631	0.450

PinRec

Table 2: CUPED-adjusted metrics improvements from online A/B experiments for PinRec variants as candidate generators in Homefeed.

Type	Metric	UC	OC	{MT, OC}
<i>Overall Metrics</i>	Fulfilled Sessions	+0.02%	+0.21%	+0.28%
	Time Spent	-0.02%	+0.16%	+0.55%
	Unfulfilled Sessions	-0.28%	-0.89%	-0.78%
<i>Action Metrics</i>	Site-wide Grid Clicks	+0.58%	+1.76%	+1.73%
	Homefeed Grid Clicks	+1.87%	+4.01%	+3.33%

PinRec

Table 3: CUPED-adjusted metrics improvements from online A/B experiments for PinRec as candidate generator in Search retrieval.

Metric Type	Metric	PinRec-UC
<i>Overall Metrics</i>	Search Fulfillment Rate	+0.48%
<i>Action Metrics</i>	Search Repins	+0.84%
	Search Grid Clicks	+1.46%

Summary лекции

Что обсудили:

- › Индустриальные тренды и где их искать, industry vs academy
- › Масштабирование
- › Actions Speak Louder Than Words, начало новой эпохи
- › LLM x RecSys
- › Foundation модели и кросс-платформенность
- › Генеративные модели (OneRec, PinRec)

Что не влезло:

- › Более структурированный и полный рассказ про LLM
- › Обучение с подкреплением
- › Mixture of logits, кандген на GPU
- › Онлайн-дообучение (e.g., тиктоковский Monolith)
- › Near-real time inference

Summary по нейросетевой части курса

- › Нейросетевое ранжирование
- › Нейросетевая генерация кандидатов
- › Графовые нейросети
- › Тренды (Генеративные рексистемы, семантические айдишники, LLM)

Спасибо

Что слушали наш курс!

... ждём вас в следующем году в гости :)