

Sequential Recommenders

Хрыльченко Кирилл (@kikhrylchenko)

ШАД 2024

- занимаюсь трансформерной персонализацией в Яндексе 3.5 года
- руковожу командой R&D рекомендательных систем 1.5 года
- внедрял модели в (Яндекс) Поиск, Рекламу, Маркет, Музыку, Кинопоиск, Алису
- выступления на datafest 2022, datafest 2023
- канал про рекомендательные системы (tg): @inforetriever

Зачем нужна трансформерная
персонализация?

Есть конечное множество **пользователей** $u \in U$ и **айтемов**¹ $i \in I$. Хотим рекомендовать пользователям айтемы, которые соответствуют их предпочтениям.

Предпочтения определяются через **фидбек**²: клики, лайки, покупки, просмотры, etc.

Хотим:

- ① много положительного фидбека в ближайшем будущем
- ② много положительного фидбека долгосрочно

¹объекты, которые мы рекомендуем

²про релевантность забудем

Пусть:

- для каждого пользователя и айтема задано **векторное представление**
- предпочтения пользователя $u \in U$ определяются через скалярное произведение $\langle u, i \rangle, i \in I$

Тогда говорят, что векторы пользователей и айтемов находятся в одном **семантическом пространстве**, а такую модель называют **двуихашенной**.

Эти векторы называют **эмбеддингами**, потому что объекты буквально вкладываются (англ. to embed) в векторное пространство.

Семантическое пространство айтемов

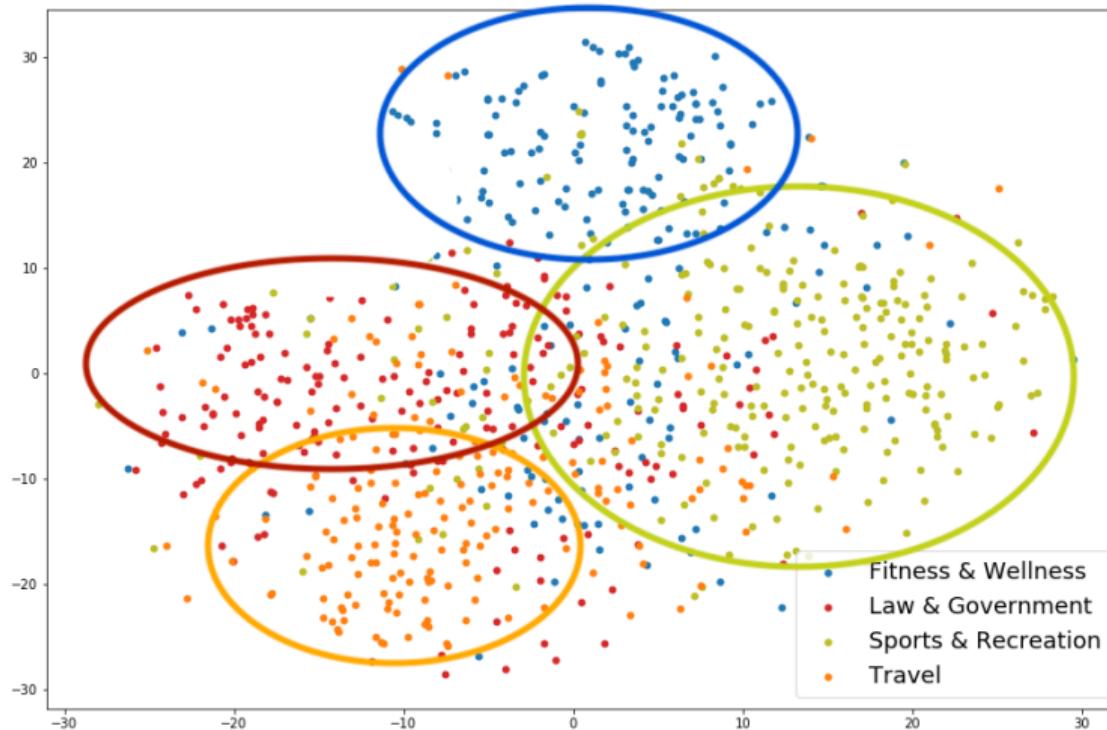


Figure: Self-supervised Learning for Large-scale Item Recommendations, 2020, Google

Почти в любой нейросети возникают векторы объектов. Иногда это целевая задача (e.g. representation learning).

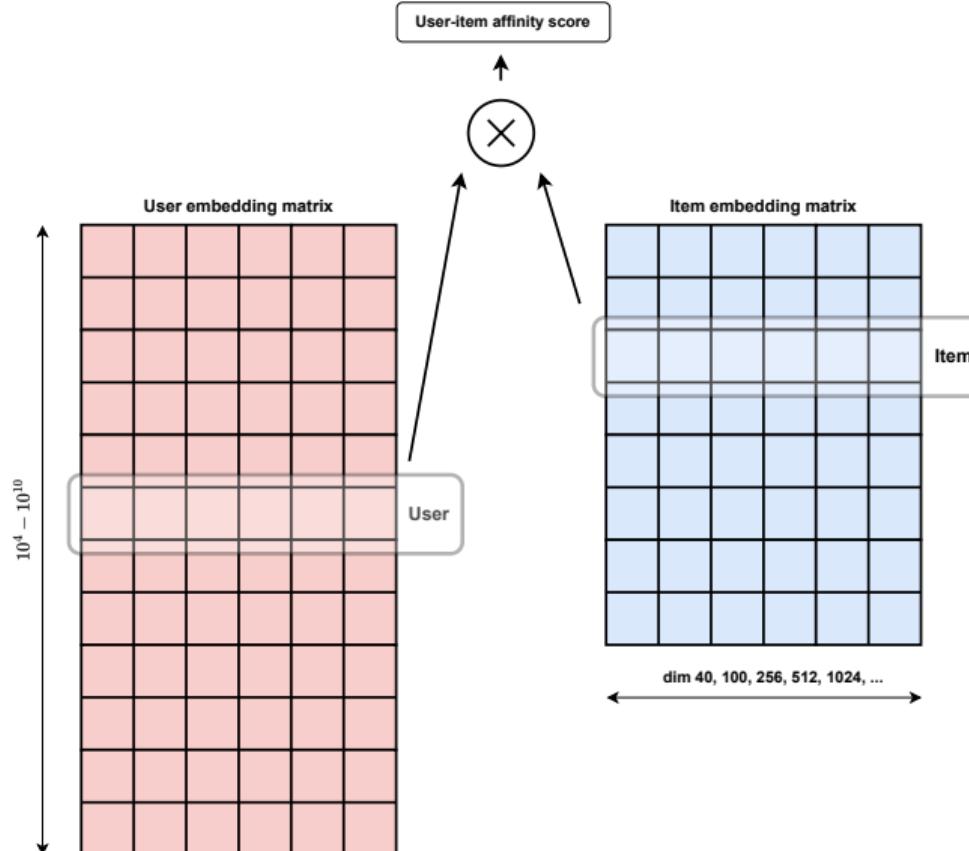
В рекомендашках:

- кандидатогенерация: **embedding-based retrieval**
- ранжирование: мощные нейросетевые признаки

В случае обучаемых эмбеддингов векторы объектов объявляются частью параметров модели; их значения подбираются в процессе оптимизации.

Рассмотрим двухбашенную модель с обучаемыми векторами: она задается функционалом потерь (+ данными для обучения), процедурой оптимизации и, непосредственно, набором обучаемых векторов.

Обучаемые векторы



У двухбашенной модели с обучаемыми векторами **максимальная емкость³** среди двухбашенных моделей.

Можно получить любую структуру семантического пространства.

Пусть у нас есть какая-то другая модель (e.g. мешок слов, трансформер, графовые нейросети), тогда можно инициализировать обучаемые эмбеддинги векторами из этой модели.

В обратную сторону это не работает - имея набор векторов, не всегда можно так настроить "content-based" модель, чтобы она выдавала нужные векторы. (вопрос: как?)

³Емкость модели (англ. model capacity) – способность выражать взаимосвязи, аппроксимировать различные функции

Проблемы рекомендательных систем, ограничивающие использование обучаемых векторов;

- недостаток данных
- тяжелые хвосты
- новые объекты
- нестационарность распределений

Data scarcity / sparsity: более сложным моделям нужно больше данных, чтобы не переобучиться (bias-variance tradeoff).

Long-tail distributions:

- Большая часть объектов мало встречается в данных
- e.g. **popularity bias**

У обучаемых эмбеддингов упор в **меморизацию** (способность запоминать обучающие данные) вместо **генерализации** (способности обобщаться на новые данные).

Обучаемые векторы **трансдуктивны**: умеют работать только на объектах, присутствующих в обучении.

Проблема **холодного старта**: большой поток новых айтемов и пользователей, поэтому нужна индуктивность – способность работать на "unseen" объектах.

В рекомендациях присутствует сильный **distribution drift**:

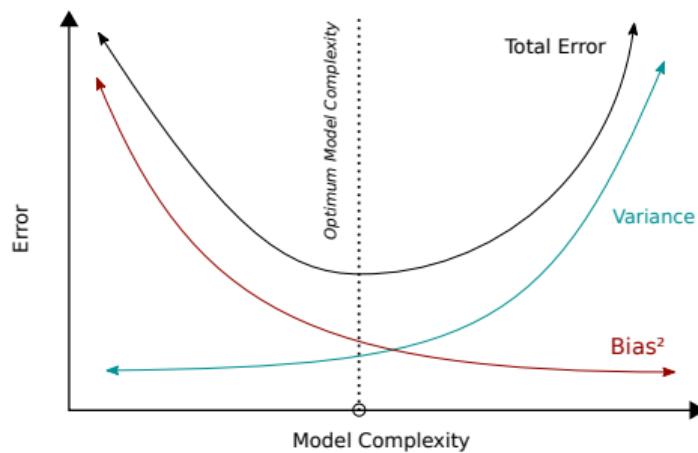
- Интересы пользователей "эволюционируют"
- Тренды (распределение популярности айтемов) меняются
- Содержимое айтемов меняется (обновляется)

Стриминговый характер данных в рекомендациях – постоянно приходит новый поток данных, на которых можно обучаться.

Инкрементальное дообучение, когда мы итеративно дообучаемся на новых данных, частично фиксирует проблемы с новыми объектами и нестационарностью, но:

- проблема курицы и яйца: чтобы дообучиться, надо накопитьフィдбек для новых объектов. Чтобы его накопить, нужно их рекомендовать
- задержка инкрементального дообучения ухудшает качество

Индуктивное смещение



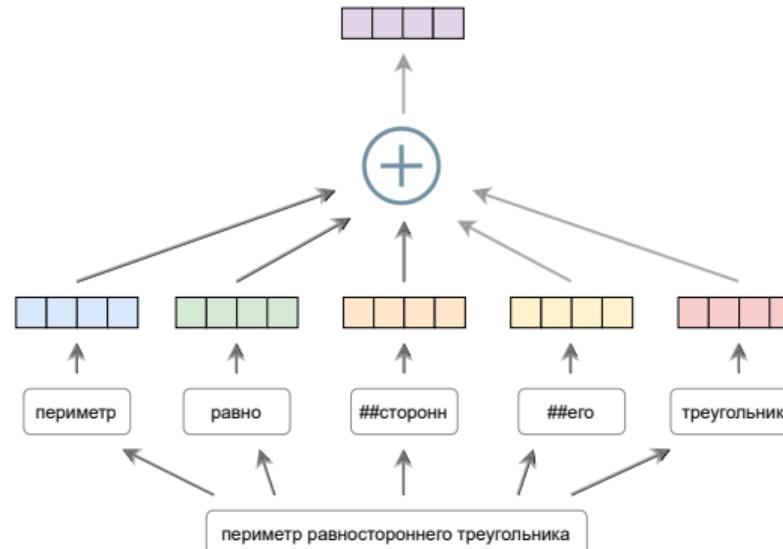
$$\text{Error} = \text{bias}^2 + \text{variance} + \text{noise}$$

Индуктивное смещение (англ. *inductive bias*) – предположения (ограничения), которые увеличивают **обобщающую способность** (англ. *generalization*). Боремся с меморизацией.

Для двухбашенных моделей: ограничиваем структуру семантического пространства.

Индуктивное смещение мешка слов

Пусть кодируем товар через **мешок слов** над названием товара:



Предположение текстового энкодера: товары с похожими названиями имеют похожую семантику с точки зрения интересов пользователя.

Для мешка слов предположение еще сильнее.

Примеры индуктивных смещений:

- ограничиваем входные данные
- ограничиваем архитектуру энкодера
- регуляризация
- нормализация эмбеддингов (перед скалярным произведением)

Добавление индуктивного смещения:

- уменьшает необходимое количество данных для генерализации
- сглаживает тяжелые хвосты: распределения фичей более "гладкие"
- облегчает холодный старт: сходу строим векторы для новых объектов по содержимому
- помогает с нестационарностью распределений: используем меняющиеся характеристики для описания объекта, фичи медленней меняются

Контент:

- текстовое описание айтема: название, описание, URL
- метаданные айтема, e.g. артист, альбом, жанр; категория товара, его характеристики
- визуальный сигнал (картинка товара, баннера)
- история взаимодействий пользователя
- описание товара через запросы, в ответ на которые на него кликали

Энкодеры: полносвязные сети, трансформеры, резнеты, пулинги, кастомные архитектуры.

- двухбашенные модели
- обучаемые эмбеддинги и их минусы
 - недостаток данных
 - тяжелые хвосты
 - холодный старт
 - нестационарность распределений
- индуктивное смещение
 - его плюсы
 - рекомендательные кейсы

Рекомендательные пулинги

Предсказать что-либо по последовательности слов (тексту) – стандартная задача в NLP. Пример: **sentiment analysis**, предсказываем тональность отзыва на фильм / ресторан / книгу.

Как анализировали текст в дотрансформерную эпоху:

- препроцессим и токенизуем текст
- каждому токену сопоставляем (обучаемый) вектор
- делаем пулинг: "агрегируем" последовательность векторов в один вектор, e.g. avgpool, maxpool, rnn, attention
- делаем предсказание из полученного вектора с помощью линейного слоя

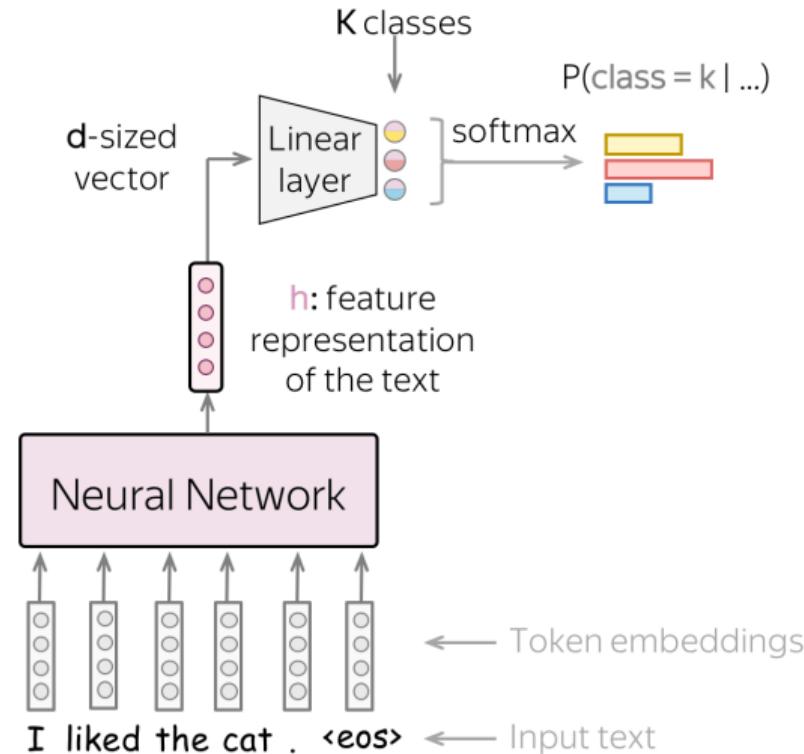


Figure: YSDA NLP Course

Двухстадийная рекомендательная система Ютуба:

- кандидатогенерация отбирает сотни видеороликов из гигантского корпуса
- ранжирование отбирает лучшие видеоролики по какой-то предсказанной характеристике, используя большой набор признаков про пользователя и айтемы

Использовали tensorflow (Google Brain).

До нейросетей использовали матричную факторизацию. Во время первые итераций нейросети заменили обучаемый вектор пользователя из факторизации на неглубокую нейросеть, формирующую эмбеддинг из прошлых просмотров.

Extreme multiclass classification:

$$P(w_i = i | U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_j u}},$$

где $u \in \mathbb{R}^n$ – эмбеддинг пары (пользователь, контекст), и $v_j \in \mathbb{R}^n$ – эмбеддинги видео-кандидатов.

На инференсе отбрасываем софтмакс и делаем **approximate nearest neighbor search**.

Вместо **explicit** фидбека (лайки, дизлайки, результаты опросов) используют **implicit** фидбек (просмотры). Досмотр видео считается позитивом. Такого фидбека гораздо больше.

*This choice is based on the orders of magnitude more implicit user history available, allowing us to produce recommendations deep in the **tail** where explicit feedback is extremely sparse.*

Сэмплируют тысячи негативов для каждого позитива и используют **importance weighting**.

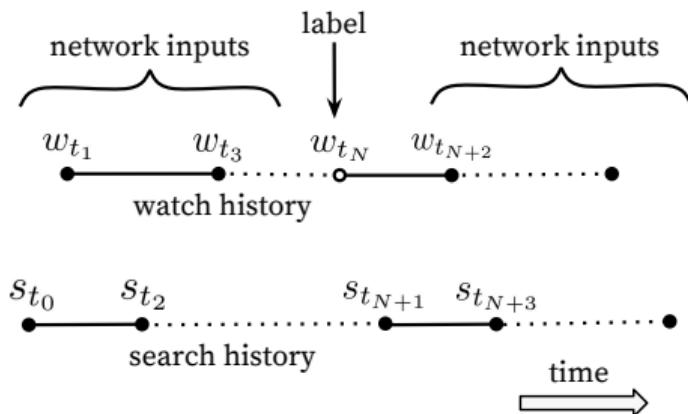
Training examples are generated from all YouTube watches (even those embedded on other sites) rather than just watches on the recommendations we produce. Otherwise, it would be very difficult for **new content** to surface and the recommender would be overly biased towards exploitation.

If users are discovering videos through means other than our recommendations, we want to be able to quickly propagate this discovery to others via collaborative filtering.

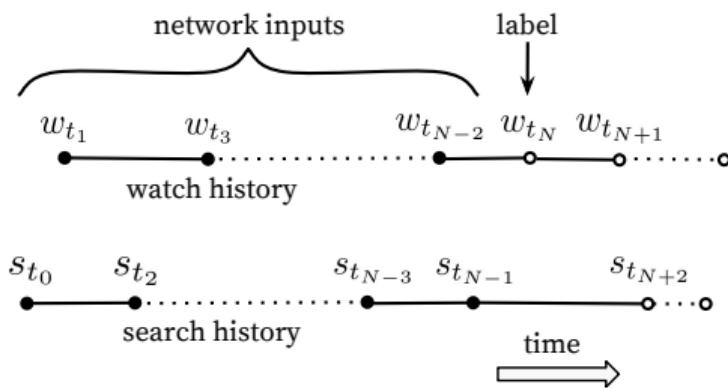
Another key insight that improved live metrics was to generate a **fixed number of training examples per user**, effectively weighting our users equally in the loss function. This prevented a small cohort of highly active users from dominating the loss.

Выбор сэмплов. Next Watch Prediction

Предсказывают следующий просмотр вместо случайного, чтобы избегать утечки информации.



(a) Predicting held-out watch



(b) Predicting future watch

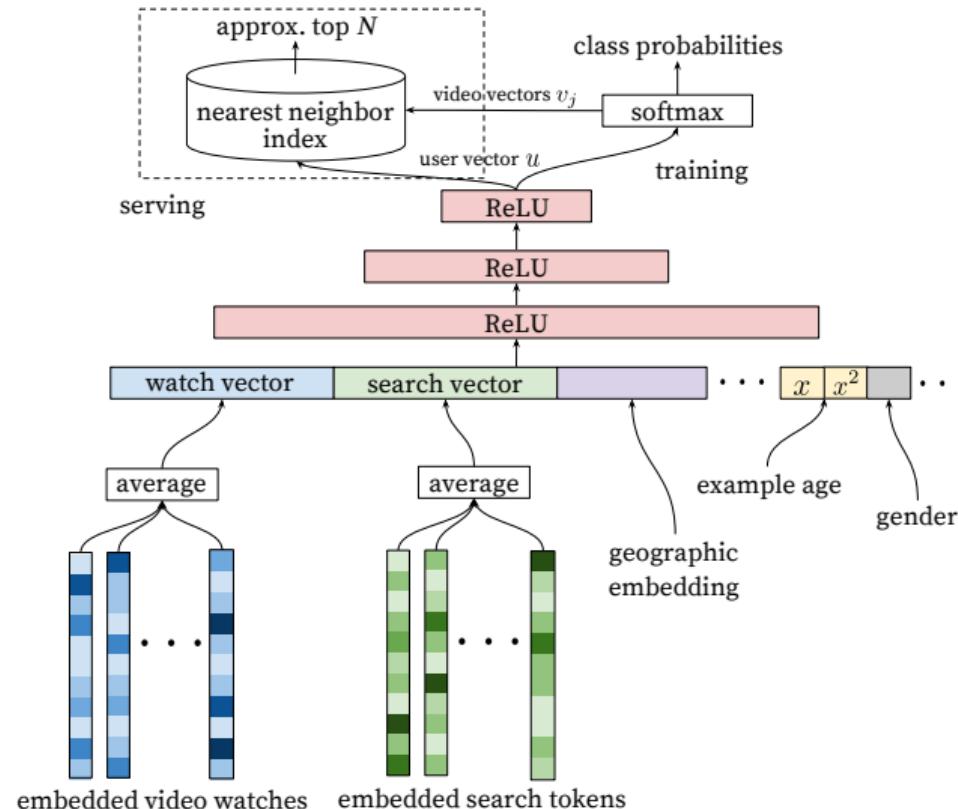
Для каждого видеоролика (из словаря фиксированного размера) заводим **обучаемый вектор**.

Эмбеддинг пользователя: эмбеддингов 50 последних просмотренных видеороликов.

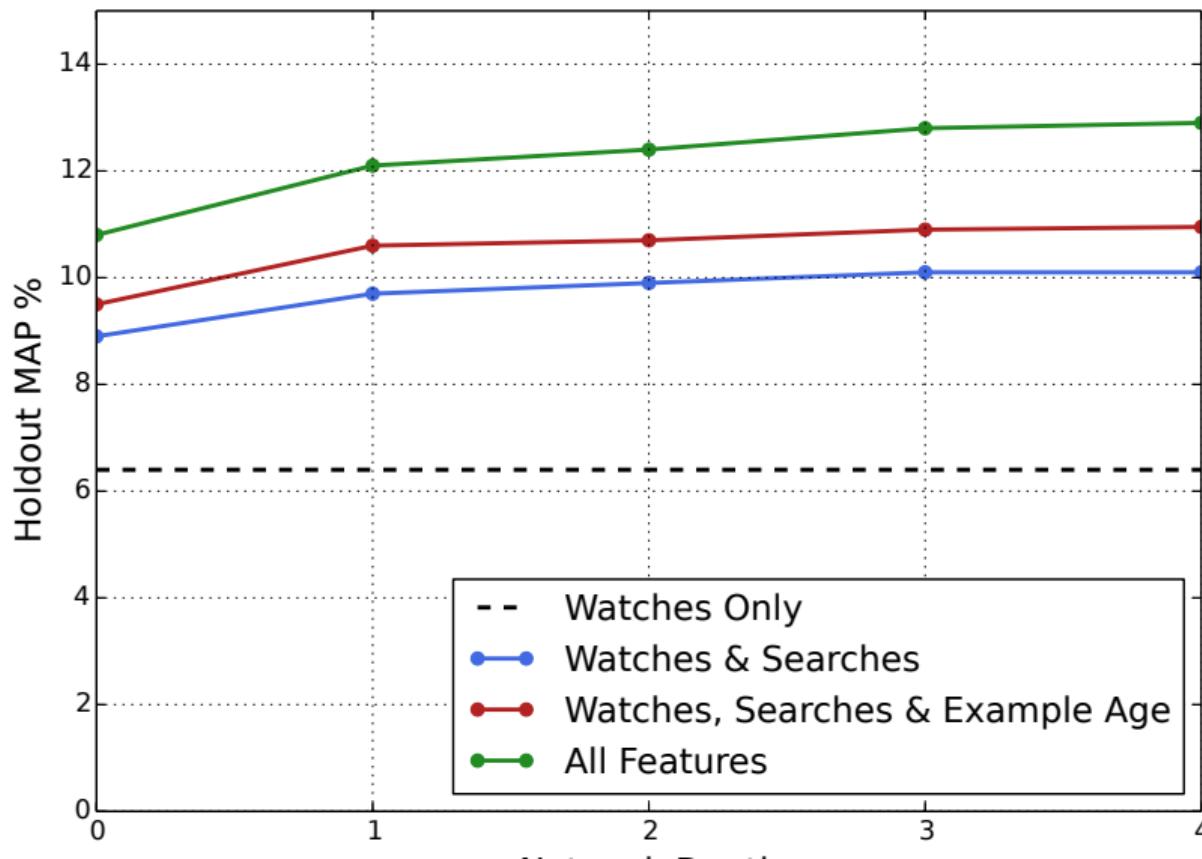
Учим эмбеддинги вместе с остальной частью нейросети, которая включает в себя конкатенацию с остальными фичами и несколько слоев с ReLU:

- **поисковый эмбед пользователья:** усреднение последних 50 запросов, представленных через униграммы и биграммы
- соц-дем признаки важны для новых пользователей
- обучаемые эмбеддинги региона и девайса
- бинарные и вещественные фичи подаются as is (user's gender, logged-in state, возраст) с нормализацией к $[0, 1]$

YoutubeDNN. Архитектура кандидатогенерации



Влияние признаков и глубины



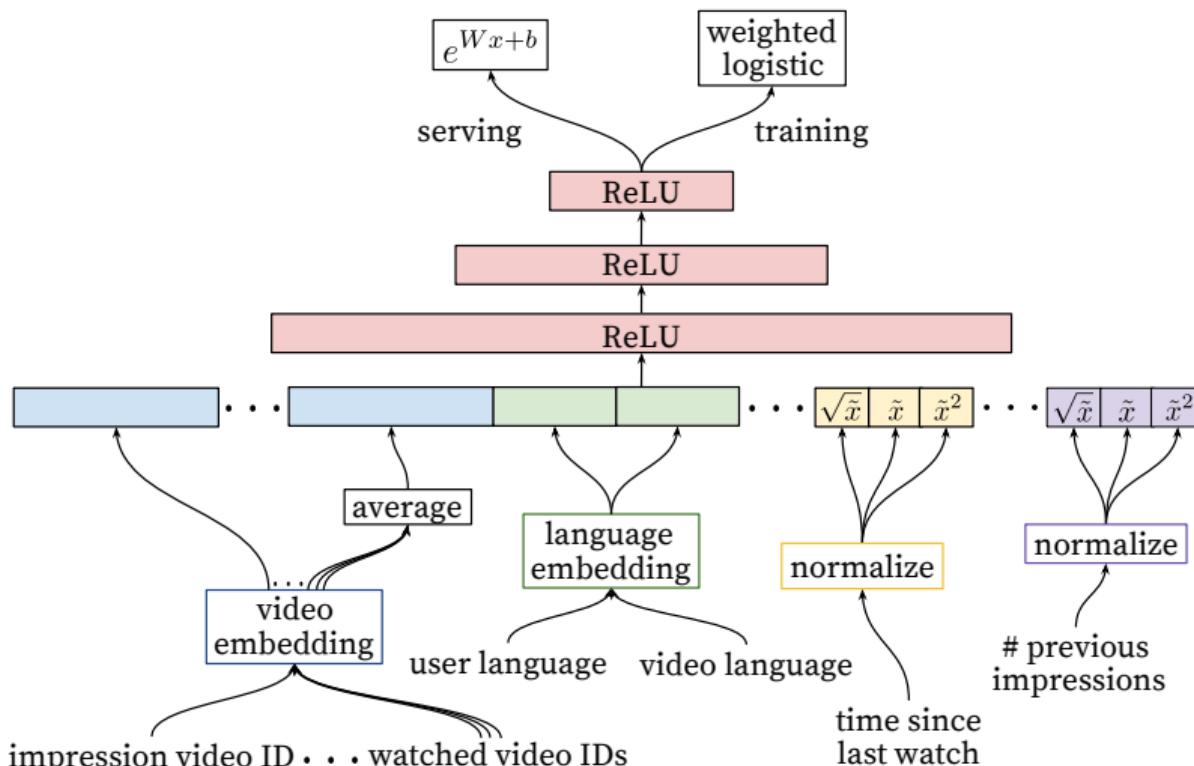
В ранжировании нужно отобрать (и отранжировать) итоговых кандидатов из пула, собранного прошлыми стадиями. Есть доступ к информации про взаимодействие айтема и пользователя – **раннее связывание**

We observe that the most important signals are those that describe a **user's previous interaction with the item itself** and other similar items, matching others' experience in ranking ads [7].

As an example, consider the user's past history with the channel that uploaded the video being scored - how many videos has the user watched from this channel? When was the last time the user watched a video on this topic?

These continuous features describing past user actions on related items are particularly powerful because they **generalize well across disparate items**.

YoutubeDNN. Архитектура ранжирования



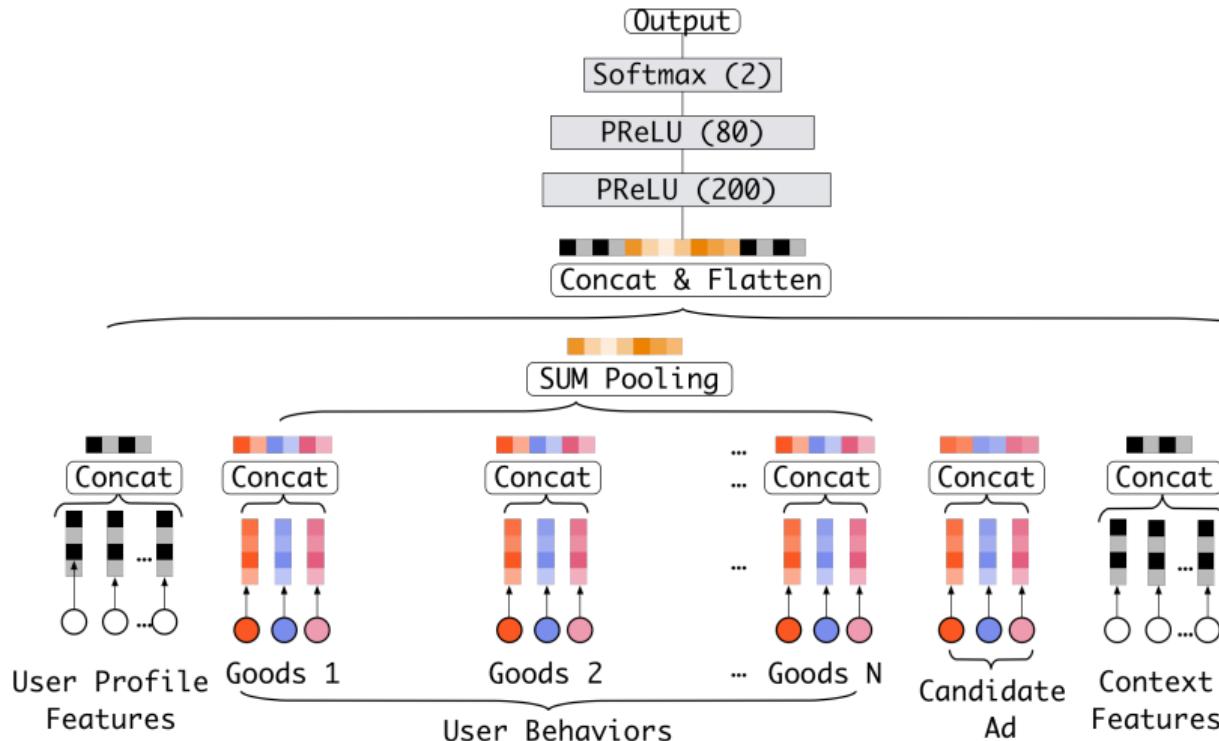
Начинали с логистической регрессии.

Display advertising в Alibaba (e-commerce):

- модель с **ранним связыванием** истории пользователя и кандидата
- предсказывали вероятность клика
- **обучаемый эмбеддинг** товара, магазина, категории товара (конкатенируют)
- mini-batch aware regularization: регуляризация только по тем айдишникам, которые попали в батч
- пробовали LSTM для анализа посл-ти, не получили профита

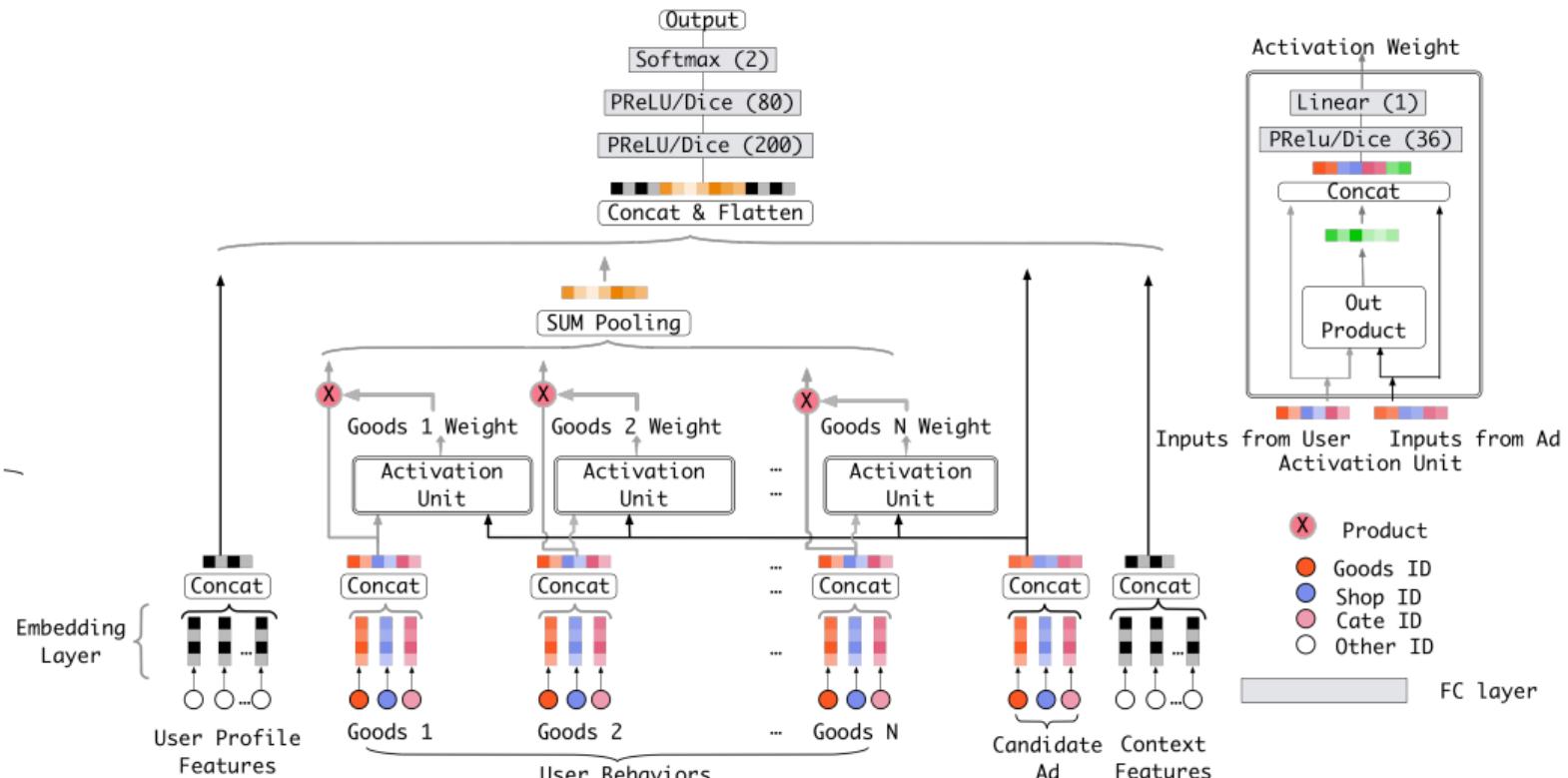
Deep Interest Network

Изначальная модель, вдохновленная YoutubeDNN:

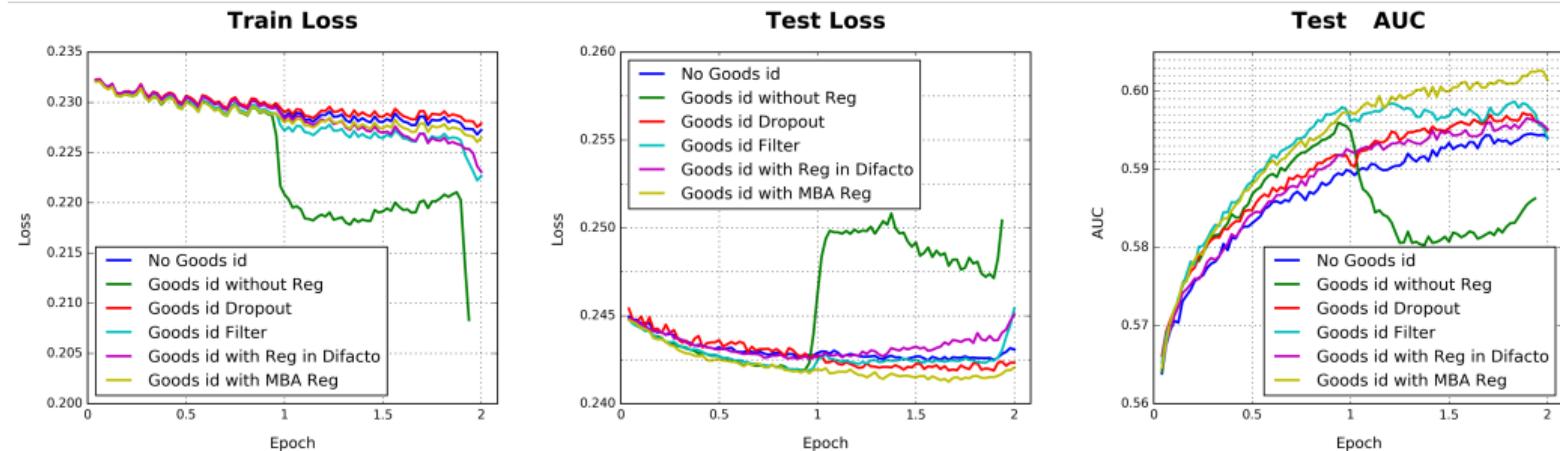


Deep Interest Network

Вместо усреднения эмбеддингов событий, один слой аттеншна с кандидатом:



Обучаемые векторы, меморизация и переобучение:



Без регуляризации начиналось сильное переобучение после первой эпохи⁴.

Вопрос: почему не делать регуляризацию сразу по всем эмбеддингам?

⁴проход по датасету

Deep Interest Network



Figure 5: Illustration of adaptive activation in DIN. Behaviors with high relevance to candidate ad get high activation weight.

Deep Interest Network

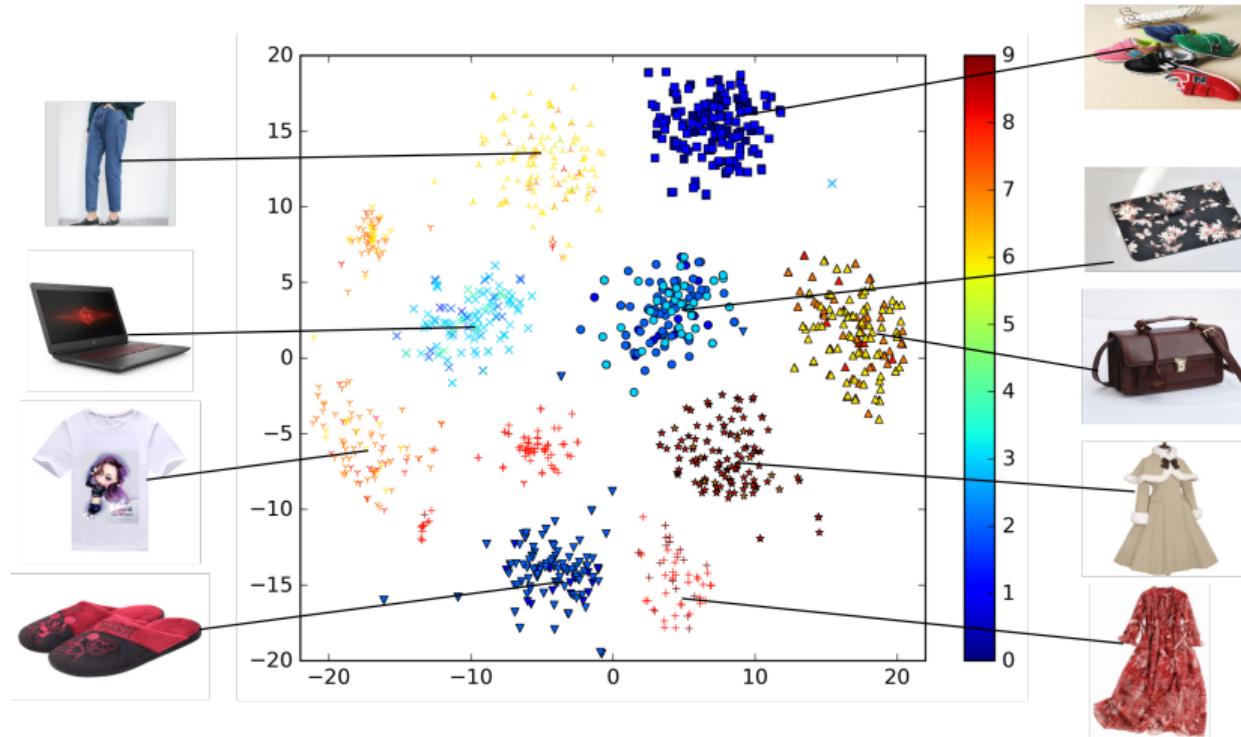


Figure 6: Visualization of embeddings of goods in DIN.

Что было после:

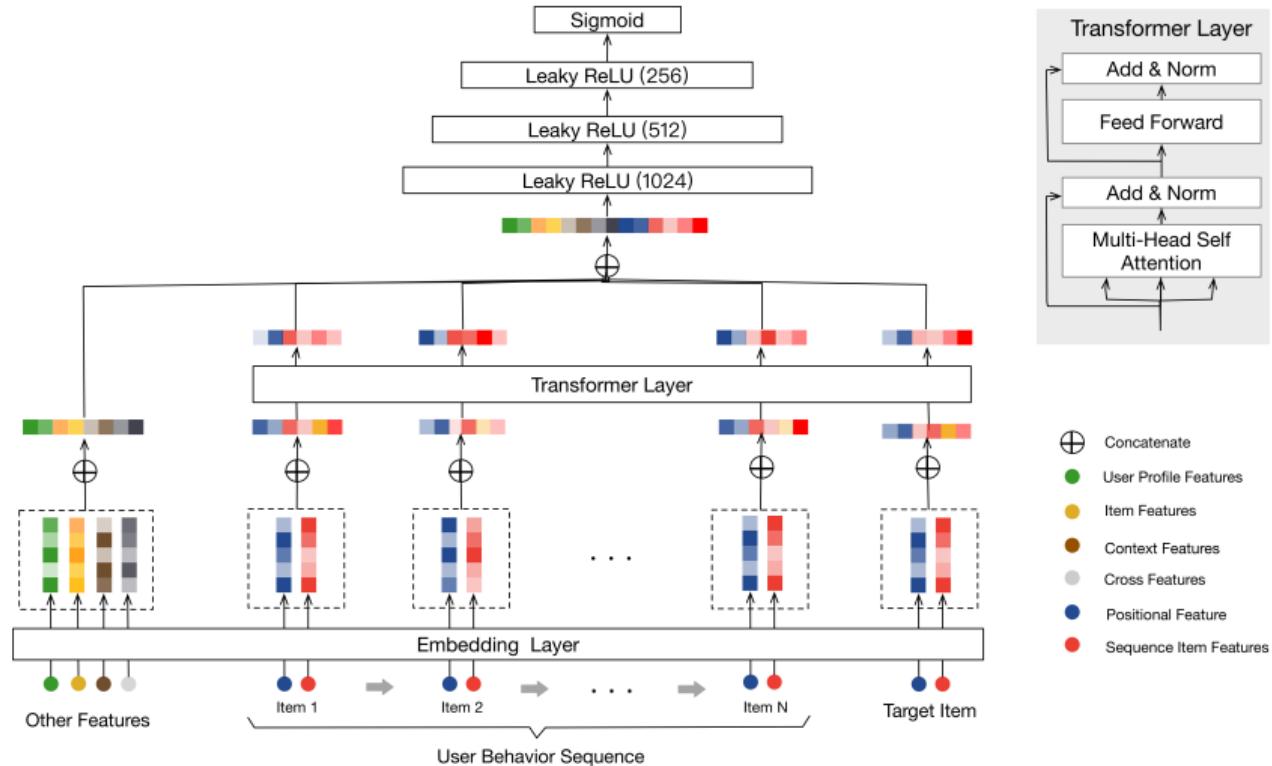
- Deep Interest Evolution Network for Click-Through Rate Prediction
- Deep Adaptive Interest Network for CTR Prediction
- Deep Context Interest Network for Click-Through Rate Prediction
- Deep Group Interest Modeling of Full Lifelong User Behaviors for CTR Prediction
- Deep Time-Aware Item Evolution Network for Click-Through Rate prediction
- Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction
- Deep Interest Highlight Network for Click-Through Rate Prediction in Trigger-Induced Recommendation
- Deep Multi-Interest Network for Click-Through Rate Prediction
- Deep Evolutional Instant Interest Network for CTR Prediction in Trigger-Induced Recommendations

Configuration of BST.

embedding size	4 ~64	batch size	256
head number	8	dropout	0.2
sequence length	20	#epochs	1
transformer block	1	queue capacity	1024
MLP Shape	1024 * 512 * 256	learning rate	0.01

Ранжирование рекомендаций в Taobao:

- заменили аттеншн из DIN на один слой **трансформера**
- несколько слоев трансформера работали хуже чем один
- для айтема используется **обучаемый вектор**, а также обучаемый вектор для категории
- добавили **позиционные эмбеддинги**: обучаемый эмбеддинг для разницы таймстемпов $pos(v_i) = t(v_t) - t(v_i)$, где $t(v_t)$ время рекомендации и $t(v_i)$ – таймстемп клика на айтем v_i ; работает лучше sin/cos
- позиционный эмбеддинг конкатенируется к эмбеддингу айтема



Methods	Offline AUC	Online CTR Gain	Average RT(ms)
WDL	0.7734	-	13
WDL(+Seq)	0.7846	+3.03%	14
DIN	0.7866	+4.55%	16
BST($b = 1$)	0.7894	+7.57%	20
BST($b = 2$)	0.7885	-	-
BST($b = 3$)	0.7823	-	-

Сравнение моделей:

- **WDL** – без анализа посл-ти
- **WDL(+Seq)** – avg пулинг над посл-тью
- **DIN** – раннее связывание (аттеншн)
- **BST** – трансформер с ранним связыванием

Next Item Prediction

Языковое моделирование – предсказываем следующее слово в тексте.

Этим занимались и в дотрансформерную эпоху: начиная с цепей Маркова, заканчивая RNN'ками.

Есть и не авторегрессивные постановки, e.g. двунаправленное языковое моделирования в BERT.

Language Modeling

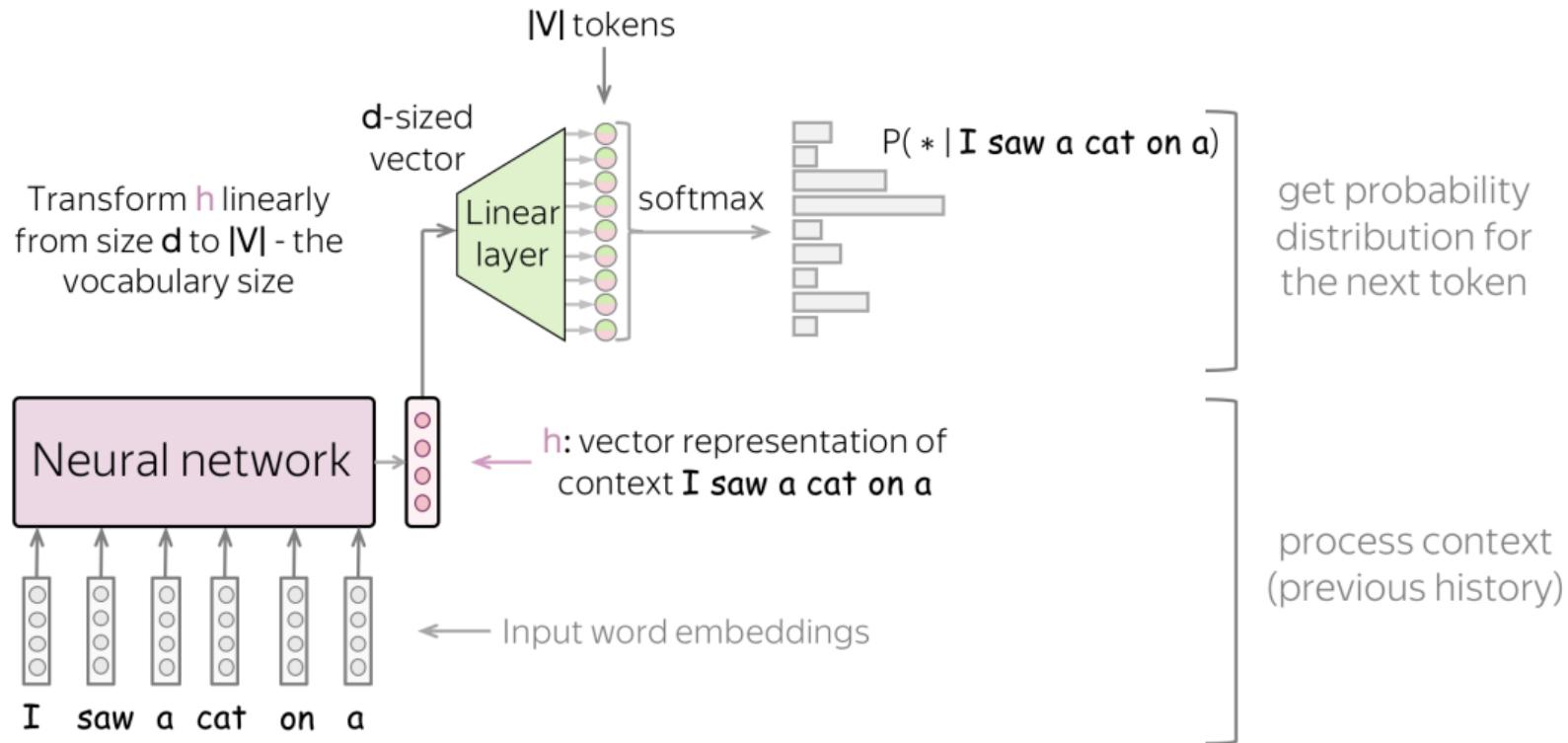


Figure: YSDA NLP Course

Language Modeling

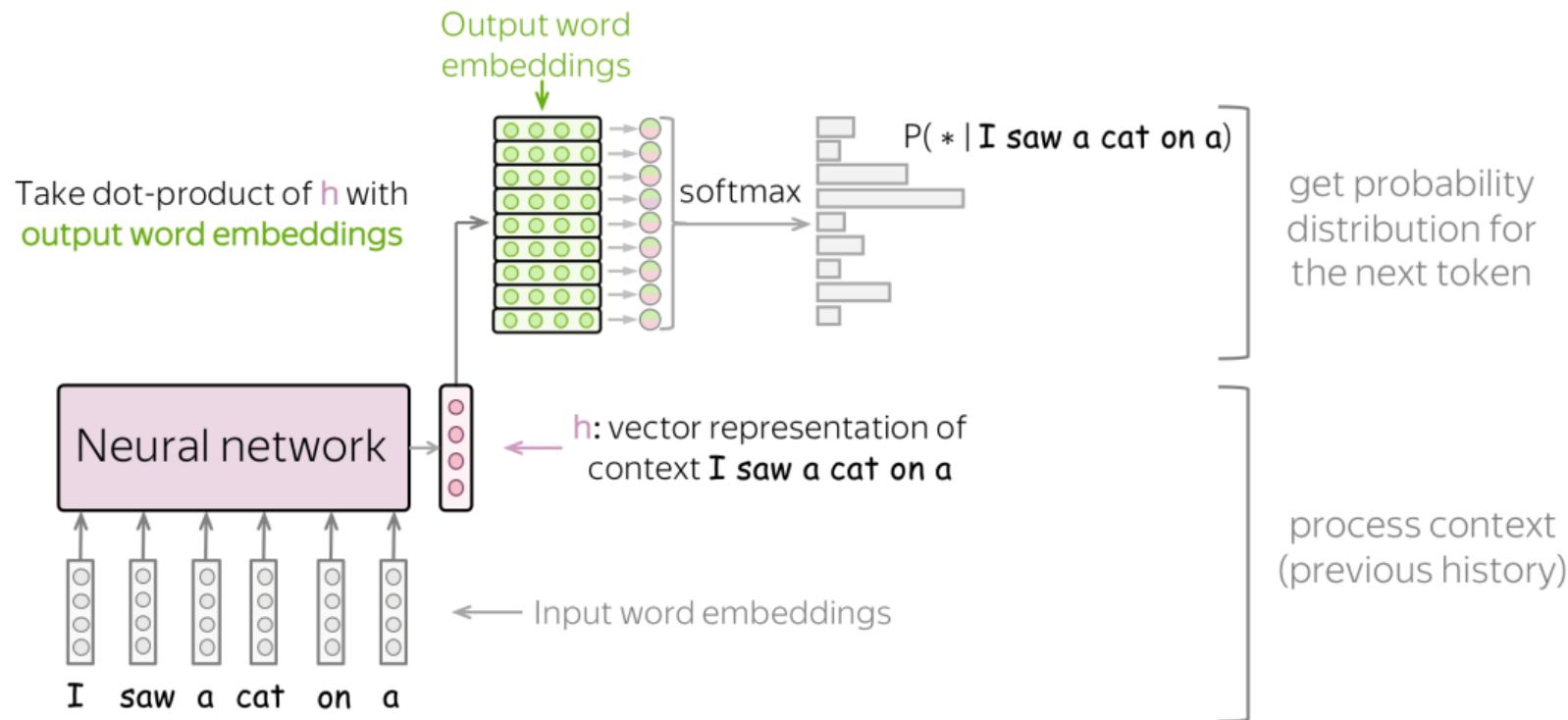
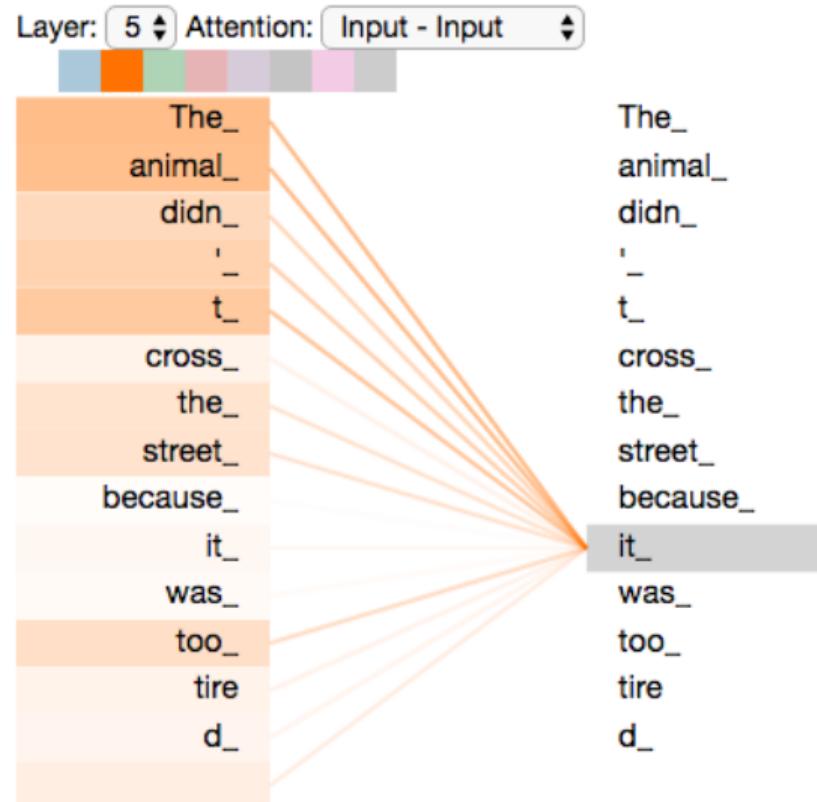
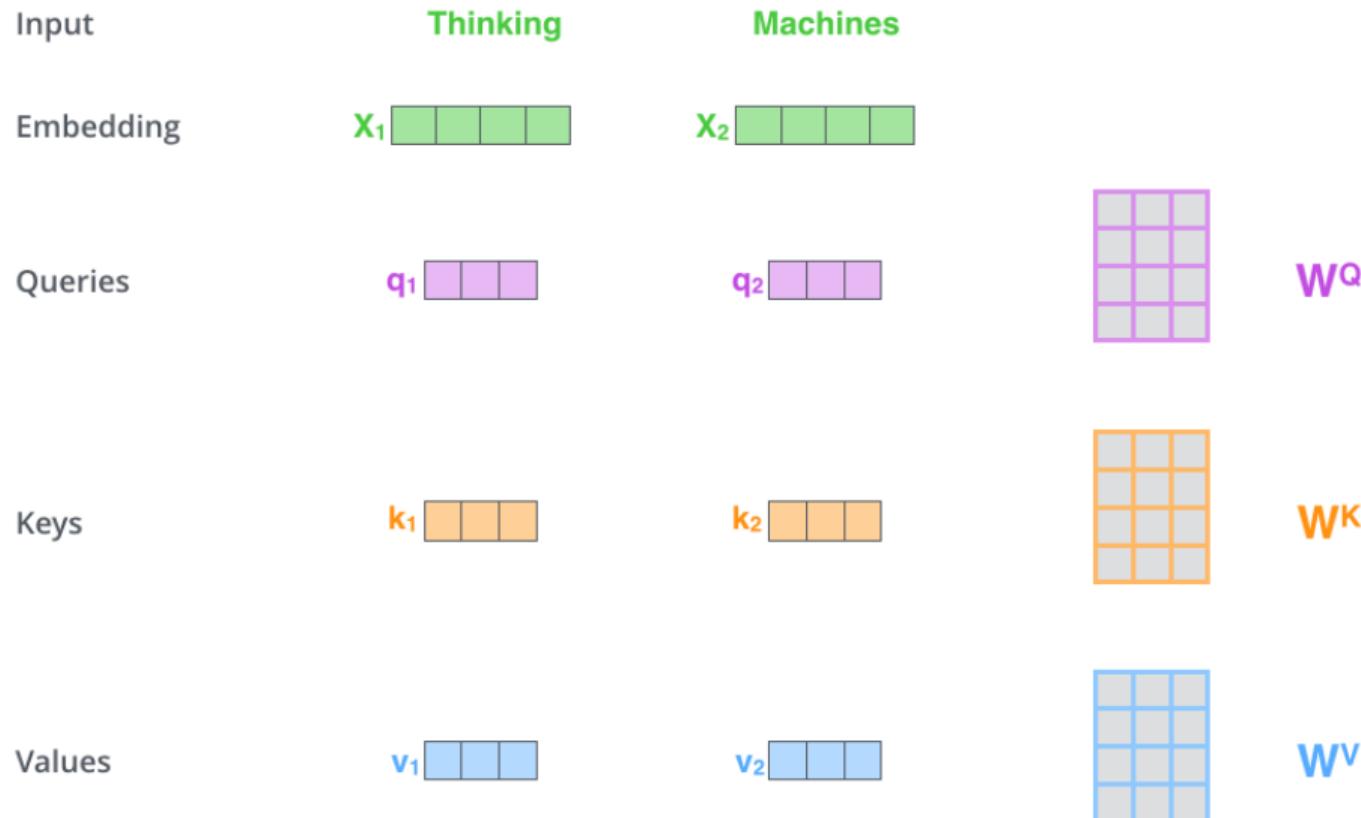


Figure: YSDA NLP Course

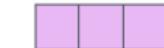
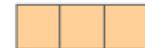
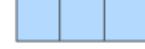
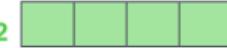
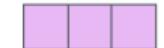
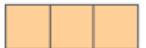
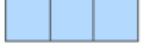
Self-attention



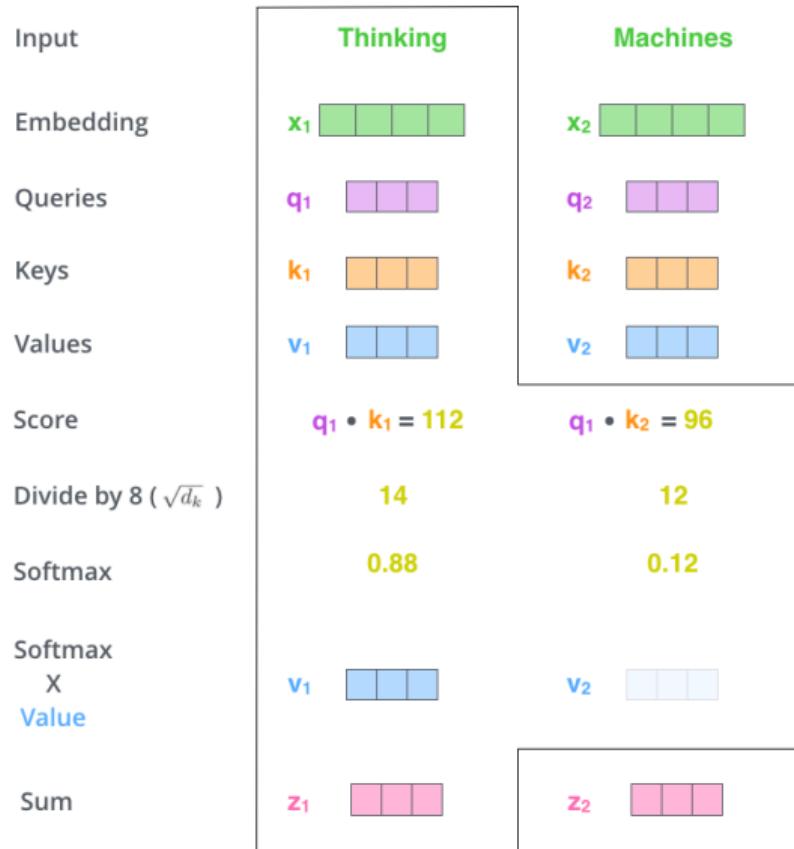
Self-attention



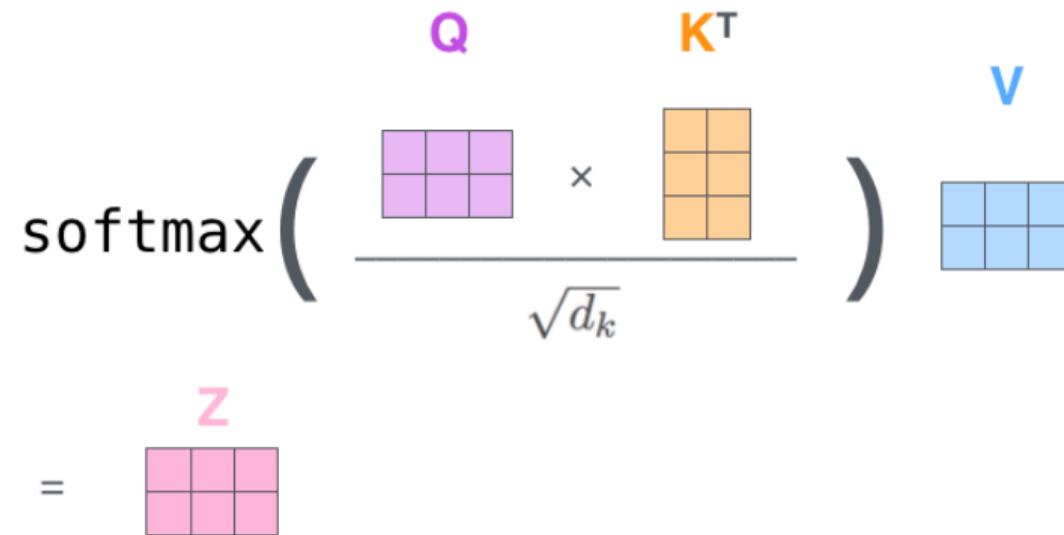
Self-attention

Input		
Embedding	x_1	
Queries	q_1	
Keys	k_1	
Values	v_1	
Score	$q_1 \cdot k_1 = 112$	
Divide by 8 ($\sqrt{d_k}$)	14	
Softmax	0.88	
Input		
Embedding	x_2	
Queries	q_2	
Keys	k_2	
Values	v_2	
Score	$q_1 \cdot k_2 = 96$	
Divide by 8 ($\sqrt{d_k}$)	12	
Softmax	0.12	

Self-attention

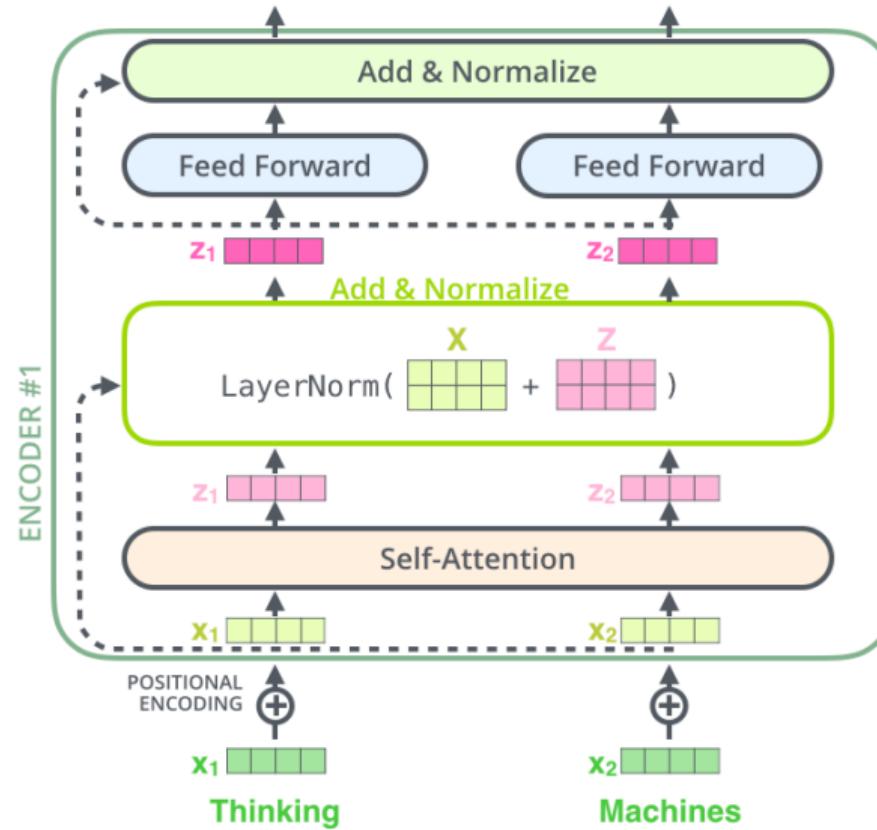


Self-attention

$$\text{softmax} \left(\frac{\begin{matrix} \mathbf{Q} & \mathbf{K}^T \\ \times & \end{matrix}}{\sqrt{d_k}} \right) \mathbf{V}$$
$$= \mathbf{Z}$$


$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}, \quad \mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{md}$$

Transformer



Полносвязный слой:

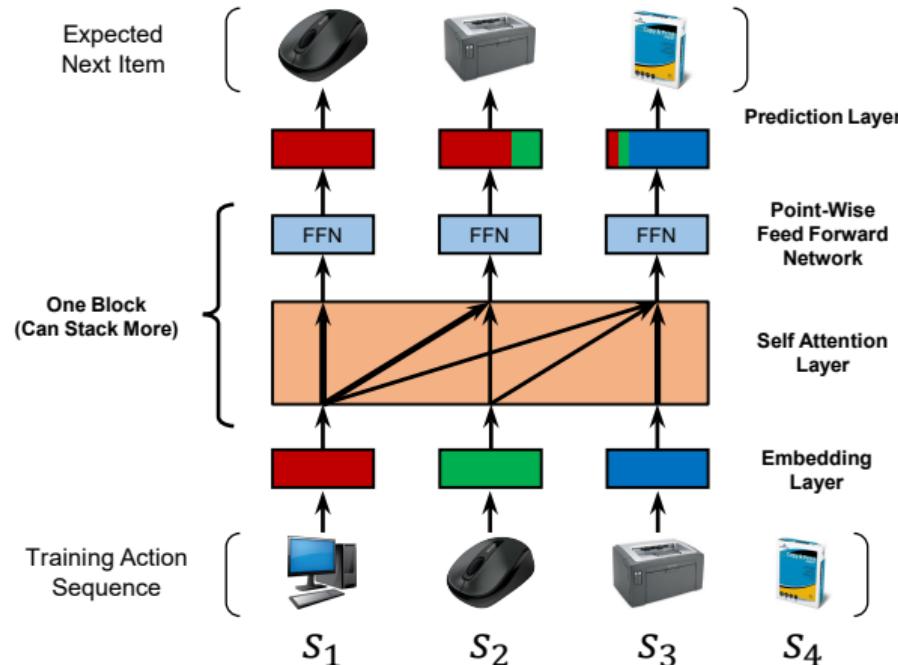
$$\text{FFN}(x) = W_2 \text{ReLU}(W_1 x + b_1) + b_2,$$

где W_1 – расширяющая (в 4 раза по дефолту).

Многоголовый аттеншн:

- делим Q, K, V на кусочки
- по каждому делаем свой аттеншн
- конкатенируем результаты и проводим через линейный слой

Next Item Prediction



Слова = айтемы, предложения = пользователи => предсказываем следующее взаимодействие.

Архитектура:

- суммируем обучаемые векторы для айтемов и для позиций
- 2 блока трансформера с авторегрессивной (треугольной) маской, одноголовый аттеншн
- чтобы предсказать следующий айтэм, переиспользуем входную матрицу обучаемых векторов айтэмов
- длина истории 50 – 200

Оптимизация:

- бинарная кросс-энтропия
- для каждого предсказываемого айтэма сэмплируем один случайный как негатив
- batch size 128, lr 1e-4, dropout 0.2 – 0.5, Adam optimizer

Dataset	#users	#items	avg. actions /user	avg. actions /item	#actions
<i>Amazon Beauty</i>	52,024	57,289	7.6	6.9	0.4M
<i>Amazon Games</i>	31,013	23,715	9.3	12.1	0.3M
<i>Steam</i>	334,730	13,047	11.0	282.5	3.7M
<i>MovieLens-1M</i>	6,040	3,416	163.5	289.1	1.0M

Оставляют только пользователей и айтемы, у которых имеется хотя бы 5 взаимодействий.

Leave-one-out схема. История пользователя делится на три части: (1) последнее действие – тест, (2) предпоследнее – валидация, (3) остальные – обучение.

Метрики: HitRate@10, NDCG@10. Сэмплируют 100 случайных айтемов для каждого пользователя и ранжируют 101 айтем.

Проблемы сетапа:

- нет деления на трейн/тест по времени. Data leakage! Меморизация вместо генерализации
- имитация логирующей политики. Более высокие метрики у моделей, похожих на прод
- предсказание следующего события не соответствует долгосрочные интересы пользователя
- сэмплирование сотни негативов не проверяет ни ранжирование, ни кандидаты. Еще и не консистентно!

SASRec. Результаты

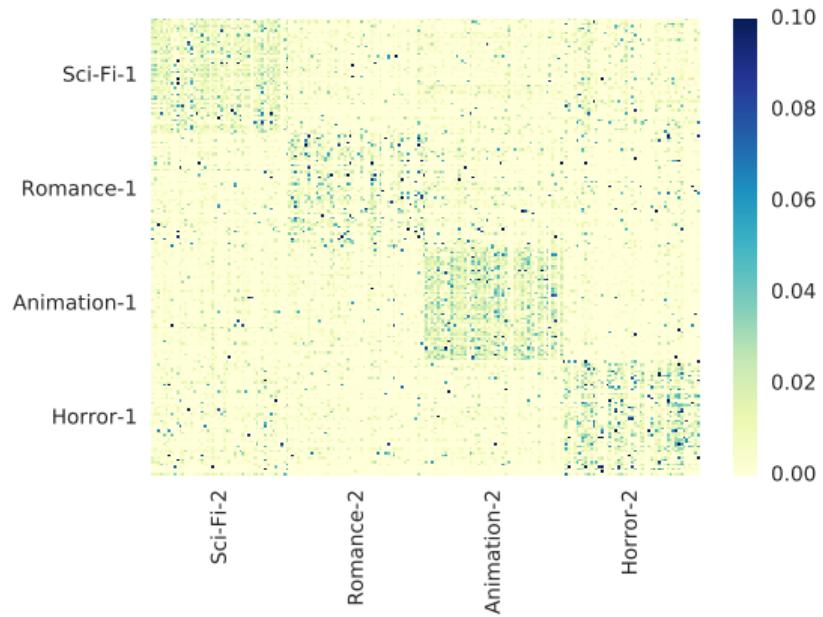
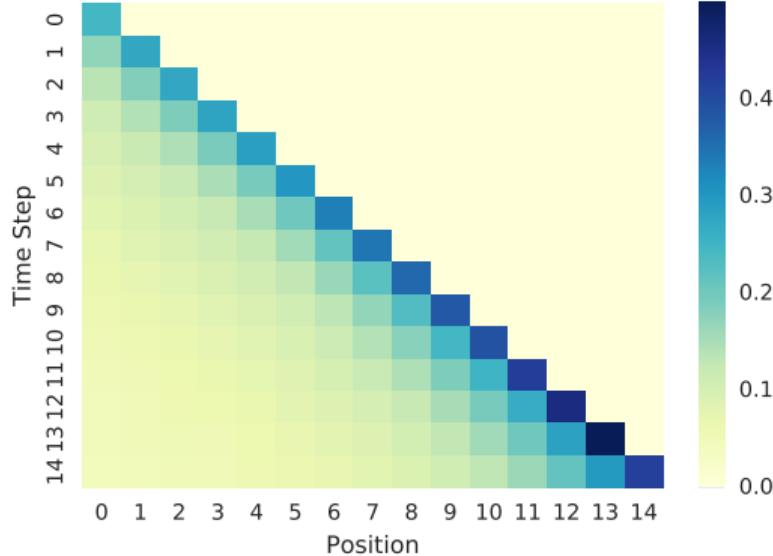
Dataset	Metric	(a) PopRec	(b) BPR	(c) FMC	(d) FPMC	(e) TransRec	(f) GRU4Rec	(g) GRU4Rec ⁺	(h) Caser	(i) SASRec	Improvement vs. (a)-(e)	Improvement vs. (f)-(h)
<i>Beauty</i>	Hit@10	0.4003	0.3775	0.3771	0.4310	<u>0.4607</u>	0.2125	0.3949	0.4264	0.4854	5.4%	13.8%
	NDCG@10	0.2277	0.2183	0.2477	0.2891	<u>0.3020</u>	0.1203	0.2556	0.2547	0.3219	6.6%	25.9%
<i>Games</i>	Hit@10	0.4724	0.4853	0.6358	0.6802	<u>0.6838</u>	0.2938	0.6599	0.5282	0.7410	8.5%	12.3%
	NDCG@10	0.2779	0.2875	0.4456	0.4680	0.4557	0.1837	<u>0.4759</u>	0.3214	0.5360	14.5%	12.6%
<i>Steam</i>	Hit@10	0.7172	0.7061	0.7731	0.7710	0.7624	0.4190	<u>0.8018</u>	0.7874	0.8729	13.2%	8.9%
	NDCG@10	0.4535	0.4436	0.5193	0.5011	0.4852	0.2691	<u>0.5595</u>	0.5381	0.6306	21.4%	12.7%
<i>ML-1M</i>	Hit@10	0.4329	0.5781	0.6986	0.7599	0.6413	0.5581	0.7501	<u>0.7886</u>	0.8245	8.5%	4.6%
	NDCG@10	0.2377	0.3287	0.4676	0.5176	0.3969	0.3381	0.5513	<u>0.5538</u>	0.5905	14.1%	6.6%

Бейзлайны: топ популярного, матричная факторизация, марковские цепи, GRU4Rec (RNN), CASER (свертки).

SASRec. Ablation study

Architecture	<i>Beauty</i>	<i>Games</i>	<i>Steam</i>	<i>ML-1M</i>
(0) Default	0.3142	0.5360	0.6306	0.5905
(1) Remove PE	0.3183	0.5301	0.6036	0.5772
(2) Unshared IE	0.2437↓	0.4266↓	0.4472↓	0.4557↓
(3) Remove RC	0.2591↓	0.4303↓	0.5693	0.5535
(4) Remove Dropout	0.2436↓	0.4375↓	0.5959	0.5801
(5) 0 Block ($b=0$)	0.2620↓	0.4745↓	0.5588↓	0.4830↓
(6) 1 Block ($b=1$)	0.3066	0.5408	0.6202	0.5653
(7) 3 Blocks ($b=3$)	0.3078	0.5312	0.6275	0.5931
(8) Multi-Head	0.3080	0.5311	0.6272	0.5885

SASRec. Attention



Каузальная маска форсирует односторонность модели:

- при формировании вектора i -го айтема смотрим только на позиции левее i
- хотим global receptive field, когда все айтемы видят все другие айтемы
- двунаправленность: при анализе айтема смотрим и в будущее, и в прошлое

Чтобы учить двунаправленную модель, нужно из одной истории длины M делать $M - 1$ сэмплов вида $([i_1], i_2), ([i_1, i_2], i_3), \dots, ([i_1, \dots, i_{M-1}], i_M)$. Это долго!

Решение: Cloze task.

Маскируем (заменяем на [MASK]) айтемы на случайных позициях в истории, затем учимся их восстанавливать.

Input: $[v_1, v_2, v_3, v_4, v_5] \xrightarrow{\text{randomly mask}} [v_1, [\text{mask}]_1, v_3, [\text{mask}]_2, v_5]$

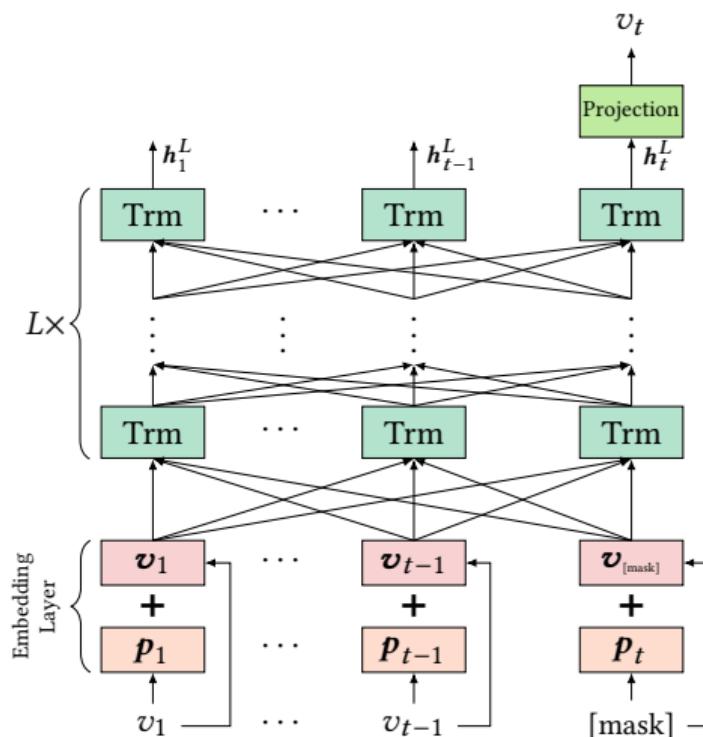
Labels: $[\text{mask}]_1 = v_2, [\text{mask}]_2 = v_4$

$$\mathcal{L} = \frac{1}{|S_u^m|} \sum_{v_m \in S_u^m} -\log P(v_m = v_m^* | S_u'),$$

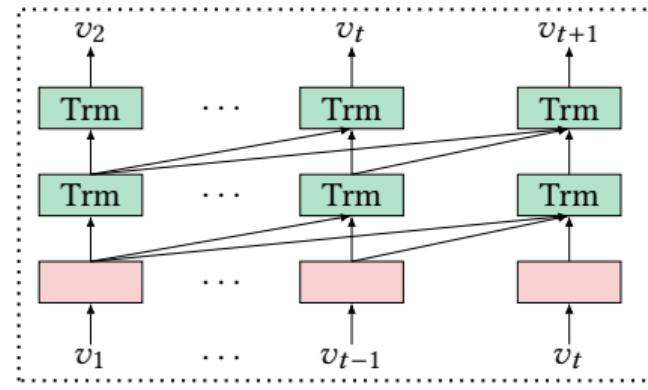
где S_u' – маскированная история пользователя u , S_u^m – маскированные айтемы.

- $P(v) = \text{softmax}(\text{GELU}(h_t^L W^P + b^P) E^T + b^O)$
- для истории длины n SASRec генерит $n - 1$ сэмпл, а BERT4Rec может C_n^k
- при применении добавляем [MASK] в конец истории

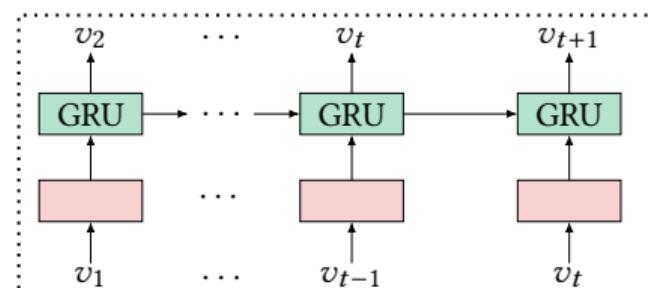
BERT4Rec



(b) BERT4Rec model architecture.



(c) SASRec model architecture.



(d) RNN based sequential recommendation methods.

Важные отличия от SASRec, которые не оцениваются:

- GELU вместо RELU
- многоголовый аттеншн (2 головы)
- сэмплируют негативы для оценки качества из униграммного распределения:
"To make the sampling reliable and representative, these 100 negative items are sampled according to their popularity."
- при обучении считают софтмакс по всем айтемам для лосса, без сэмплирования

BERT4Rec. Результаты

Datasets	Metric	POP	BPR-MF	NCF	FPMC	GRU4Rec	GRU4Rec ⁺	Caser	SASRec	BERT4Rec	Improv.
Beauty	HR@1	0.0077	0.0415	0.0407	0.0435	0.0402	0.0551	0.0475	0.0906	0.0953	5.19%
	HR@5	0.0392	0.1209	0.1305	0.1387	0.1315	0.1781	0.1625	0.1934	0.2207	14.12%
	HR@10	0.0762	0.1992	0.2142	0.2401	0.2343	0.2654	0.2590	0.2653	0.3025	14.02%
	NDCG@5	0.0230	0.0814	0.0855	0.0902	0.0812	0.1172	0.1050	0.1436	0.1599	11.35%
	NDCG@10	0.0349	0.1064	0.1124	0.1211	0.1074	0.1453	0.1360	0.1633	0.1862	14.02%
	MRR	0.0437	0.1006	0.1043	0.1056	0.1023	0.1299	0.1205	0.1536	0.1701	10.74%
Steam	HR@1	0.0159	0.0314	0.0246	0.0358	0.0574	0.0812	0.0495	0.0885	0.0957	8.14%
	HR@5	0.0805	0.1177	0.1203	0.1517	0.2171	0.2391	0.1766	0.2559	0.2710	5.90%
	HR@10	0.1389	0.1993	0.2169	0.2551	0.3313	0.3594	0.2870	0.3783	0.4013	6.08%
	NDCG@5	0.0477	0.0744	0.0717	0.0945	0.1370	0.1613	0.1131	0.1727	0.1842	6.66%
	NDCG@10	0.0665	0.1005	0.1026	0.1283	0.1802	0.2053	0.1484	0.2147	0.2261	5.31%
	MRR	0.0669	0.0942	0.0932	0.1139	0.1420	0.1757	0.1305	0.1874	0.1949	4.00%
ML-1m	HR@1	0.0141	0.0914	0.0397	0.1386	0.1583	0.2092	0.2194	0.2351	0.2863	21.78%
	HR@5	0.0715	0.2866	0.1932	0.4297	0.4673	0.5103	0.5353	0.5434	0.5876	8.13%
	HR@10	0.1358	0.4301	0.3477	0.5946	0.6207	0.6351	0.6692	0.6629	0.6970	4.15%
	NDCG@5	0.0416	0.1903	0.1146	0.2885	0.3196	0.3705	0.3832	0.3980	0.4454	11.91%
	NDCG@10	0.0621	0.2365	0.1640	0.3439	0.3627	0.4064	0.4268	0.4368	0.4818	10.32%
	MRR	0.0627	0.2009	0.1358	0.2891	0.3041	0.3462	0.3648	0.3790	0.4254	12.24%

BERT4Rec. Ablation study

Architecture	Dataset			
	Beauty	Steam	ML-1m	ML-20m
$L = 2, h = 2$	0.1832	0.2241	0.4759	0.4513
w/o PE	0.1741	0.2060	0.2155↓	0.2867↓
w/o PFFN	0.1803	0.2137	0.4544	0.4296
w/o LN	0.1642↓	0.2058	0.4334	0.4186
w/o RC	0.1619↓	0.2193	0.4643	0.4483
w/o Dropout	0.1658	0.2185	0.4553	0.4471
1 layer ($L = 1$)	0.1782	0.2122	0.4412	0.4238
3 layers ($L = 3$)	0.1859	0.2262	0.4864	0.4661
4 layers ($L = 4$)	0.1834	0.2279	0.4898	0.4732
1 head ($h = 1$)	0.1853	0.2187	0.4568	0.4402
4 heads ($h = 4$)	0.1830	0.2245	0.4770	0.4520
8 heads ($h = 8$)	0.1823	0.2248	0.4743	0.4550

Что еще пробуют делать в NIP:

- менять постановку обучения, e.g. gSASRec
- использовать таймстемпы, e.g. TiSASRec
- добавлять contrastive learning с аугментациями истории, e.g. CL4SRec
- менять архитектуру энкодера, e.g. GRU4Rec, LRURec, ConvFormer
- использовать контекст и контент айтемов, e.g. CARCA
- использовать диффузионки, e.g. Diff4Rec
- прикручивать языковые модели, e.g. GPTRec, LLMRec

Next time prediction:

- industrial sequential recommenders (Pinterest, Etsy?, eBay?)
- кодирование посл-тей
- кодирование событий
- как обучать
- как применять
- lifelong recommendations (анализ тысяч событий, многовекторные модели)
- использование контекста
- LLM, RL
- универсальные эмбеддинги пользователя

Industrial Sequential Recommenders

PinnerSage

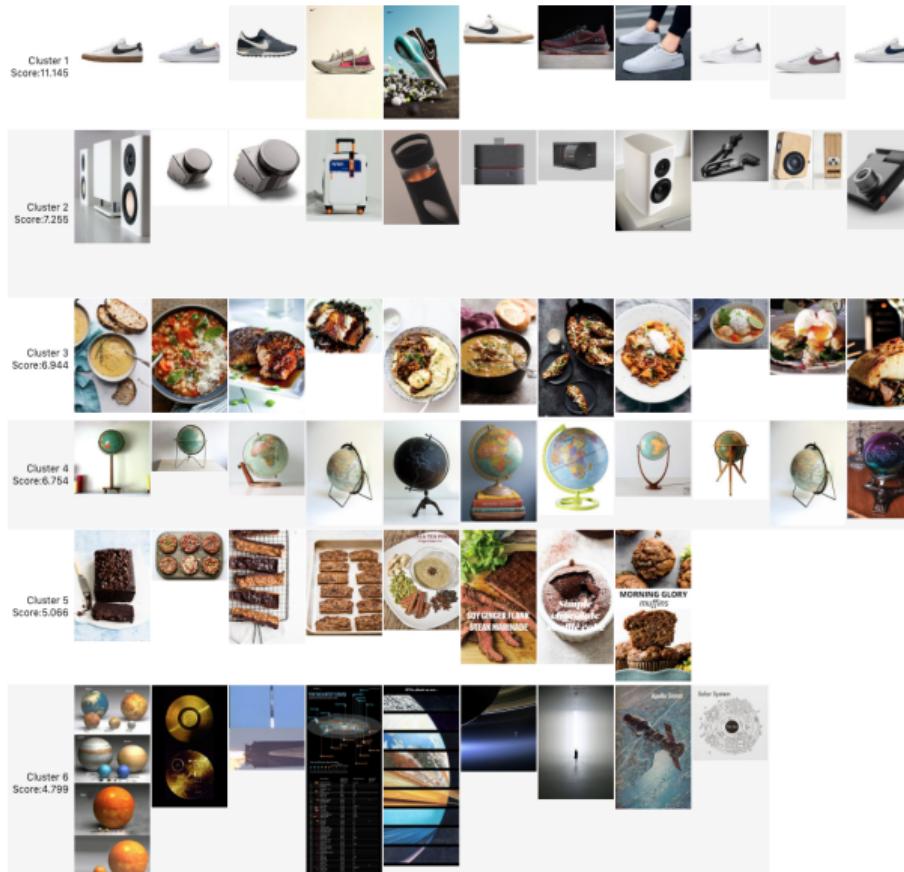


Figure 5: PinnerSage clusters of an anonymous user.

Design choices:

- учат векторное представление для пользователя по его истории взаимодействий с пинами
- *upstream* модель: единое векторное представление пользователя, переиспользуемое для *downstream* моделей
- один эмбеддинг пользователя вместо нескольких: проще для переиспользования
- *offline* модель вместо *realtime*: нужно меньше ресурсов и инфраструктура проще

PinSage эмбеддинг – "гибридный" вектор пина размерности 256, агрегирующий информацию про текст, картинку и пользовательских фидбек (графовая нейросеть).

Кодируем метаданные взаимодействия пользователя с пином:

- тип действия (save, close-up, click) и поверхность (главная страница, поиск, похожие пины) кодируем обучаемым эмбеддингом
- логарифм от длительности действия
- timestamp-based эмбеддинги: время до последнего действия и время между действиями; cos, sin с набором фиксированных периодов, логарифм значения
- конкatenируем все в единый вектор

Модель двухбашенная: отдельно формируем эмбеддинги пользователей и айтемов.

Энкодер пользователя:

- **история пользователя:** самые свежие M взаимодействий с пинами (клики, сохранения, реакции, комменты) за последний год
- к эмбеддингам исторических событий добавляются **обучаемые позиционные эмбеддинги**
- **трансформер с prenorm** в качестве энкодера
- небольшой MLP над выходами трансформера

Энкодер пина: MLP над PinSage эмбеддингом.

Оба эмбеддинга l2-нормализуются (для стабильности), то есть в качестве метрики используется косинус.

Для обучения нужно отобрать пары $\{(u_1, p_1), \dots, (u_B, p_B)\}$, состоящие из эмбеддингов пользователей и айтемов, которые при этом потенциально могут повторяться:

- как составить пары?
- как выбрать негативные примеры для пары (u_i, p_i) ?
- имея пару (u_i, p_i) и набор негативов, какую функцию потерь считать?

Явные (explicit) негативы не используют.

Вопрос: что такое explicit негативы? А implicit негативы?

In-batch негативы – чужие позитивы, попавшие в батч.

- штрафуем популярные объекты, так как они чаще попадаются как негативы
- распределение негативов (униграмное) не совпадает с реальным распределением айтемов при применении (равномерное)

Случайные негативы равно сэмплируются из всего каталога. Как единственный источник они слишком простые, могут вызвать model collapse.

Mixed negative sampling – используем и то, и другое.

Собирают негативы со всех карточек (cross-device).

Loss Function

Для пары (u_i, p_i) и отобранных негативов $\{n_1, \dots, n_N\}$ считаем **sampled softmax loss** с **logQ** коррекцией:

$$\mathcal{L}(u_i, p_i) = -\log \left(\frac{e^{s(u_i, p_i) - \log(Q_i(p_i))}}{e^{s(u_i, p_i) - \log(Q_i(p_i))} + \sum_{j=1}^N e^{s(u_i, n_j) - \log(Q_i(n_j))}} \right),$$

где:

- $s(u, p) = \langle p, u \rangle / \tau$
- $\tau \in [0.01, \inf)$ – обучаемая температура
- $Q_i(v) = P(\text{Pin } v \text{ in batch} \mid \text{User } u_i \text{ in batch})$ считается с помощью count-min sketch

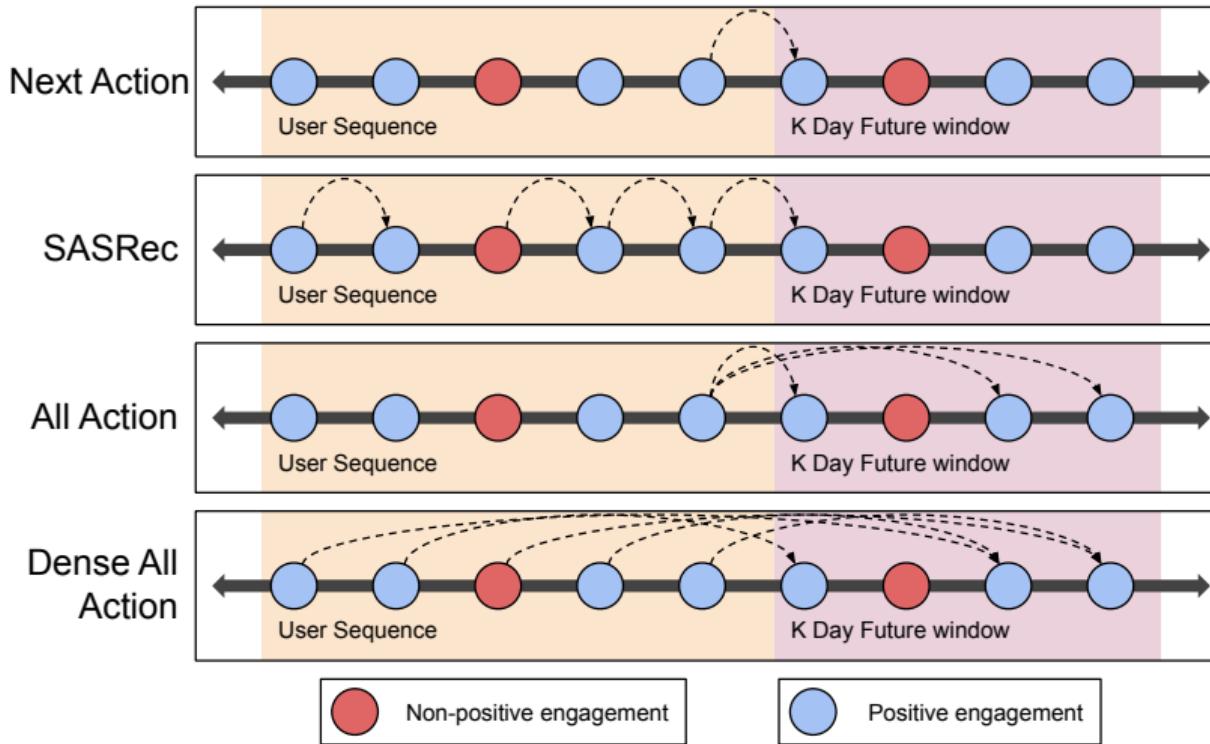
Лосс усредняется по пользователям.

Позитивы: repin, closeup, click на главной странице, без перевзвешивания.

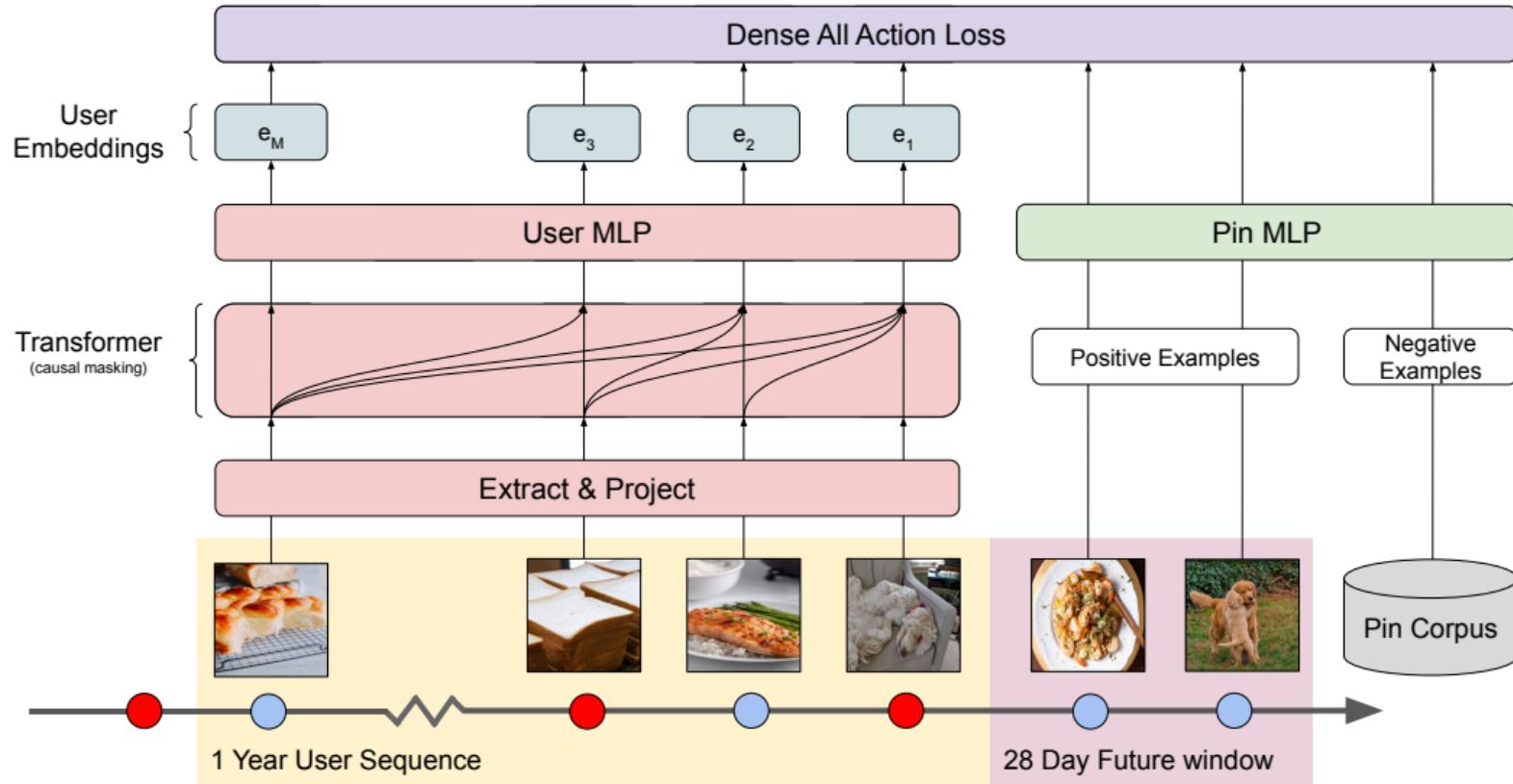
Задачи обучения:

- **next action prediction** (non-causal), SASRec (causal + teacher forcing)
- **all action prediction** (долгосрочный сигнал)
 - предсказываем до 32-х сэмплированных позитивов пользователя за следующие K дней (K=14)
 - используем последний вектор пользователя
- **dense all action** (вдохновлено SASRec)
 - предсказываем из сэмплированного набора позиций
 - предсказываем по одному позитиву из K дней с каждой
 - авторегрессивная маска в аттеншне

Training Objective



Pinnerformer



Ежедневная оффлайновая, батч операция для инференса пользовательских эмбеддингов:

- только для пользователей, у которых за последний день появились новые события
- нет ограничений на latency
- в случае порчи данных эмбеддинги легко пересчитываются

Вычисление эмбеддингов пинов-кандидатов очень легкое (MLP над готовыми PinSage эмбеддингами), каждый день пересчитывают с нуля и строят HNSW индекс.

Оценка качества

Откладываем две недели после обучения, используем только пользователей, которых не было в обучении.

Вычисляем эмбеддинги пользователей на момент начала теста.

Строим индекс из одного миллиона случайных пинов и ходим в него с этими эмбеддингами. Смотрим на $\text{Recall}@k$ для позитивов пользователя за ближайшие две недели.

Метрики разнообразия:

- **Entropy@50:** энтропия распределения тем пинов в топ 50. Разнообразие для пользователя
- **P90 Coverage@10:** покрытие индекса 90% самых частых айтемов из верхних 10 позиций выдачи. Глобальное разнообразие

Вопрос: какие значения метрик будут у топа популярного?

Сравнение с

другими моделями

Model	R@10	Interest Entropy@50	P90 Coverage@10
PS (5 clusters)	0.026	1.69	0.130
PS (20 clusters)	0.046	2.10	0.133
PINNERFORMER	0.229	1.97	0.042

Pinnerformer. Результаты

Inference Frequency	Model	R@10	P90 Coverage@10
Once	SASRec	0.198	0.048
	PINNERFORMER	0.229	0.042
Daily	SASRec	0.216	0.052
	PINNERFORMER	0.243	0.043
Realtime	SASRec	0.251	0.057
	PINNERFORMER	0.264	0.045

- Once – формируем эмбеддинги для пользователей один раз, перед тестом
- Daily – эмулируем ежедневный пересчет эмбеддингов
- Realtime – пересчитываем эмбеддинг после каждого события

Сравнение задач обучения:

Training Objective	Recall@10	P90 Coverage@10
Next Action	0.186	0.050
SASRec (Softmax)	0.198	0.048
All Action (28d)	0.224	0.028
Dense All Action (14d)	0.223	0.043
Dense All Action (28d)	0.229	0.042

Разные схемы отбора негативов:

SPC	Negative Source	P90 Coverage@10	Recall@10
N	random	0.002	0.136
N	in-batch	0.163	0.071
N	mixed	0.083	0.138
Y	random	0.001	0.139
Y	in-batch	0.119	0.167
Y	mixed	0.042	0.229

Pinnerformer. Результаты

Ablation study фичей:

Omitted Feature	P90 Coverage@10	Recall@10
PinSage	0.0005	0.142
Timestamp	0.050	0.210
Surface	0.040	0.224
Action Type	0.042	0.226
Duration	0.042	0.226
Positional Encoding	0.041	0.228
None	0.042	0.229

Pinnerformer. Результаты

Ablation study длины истории:

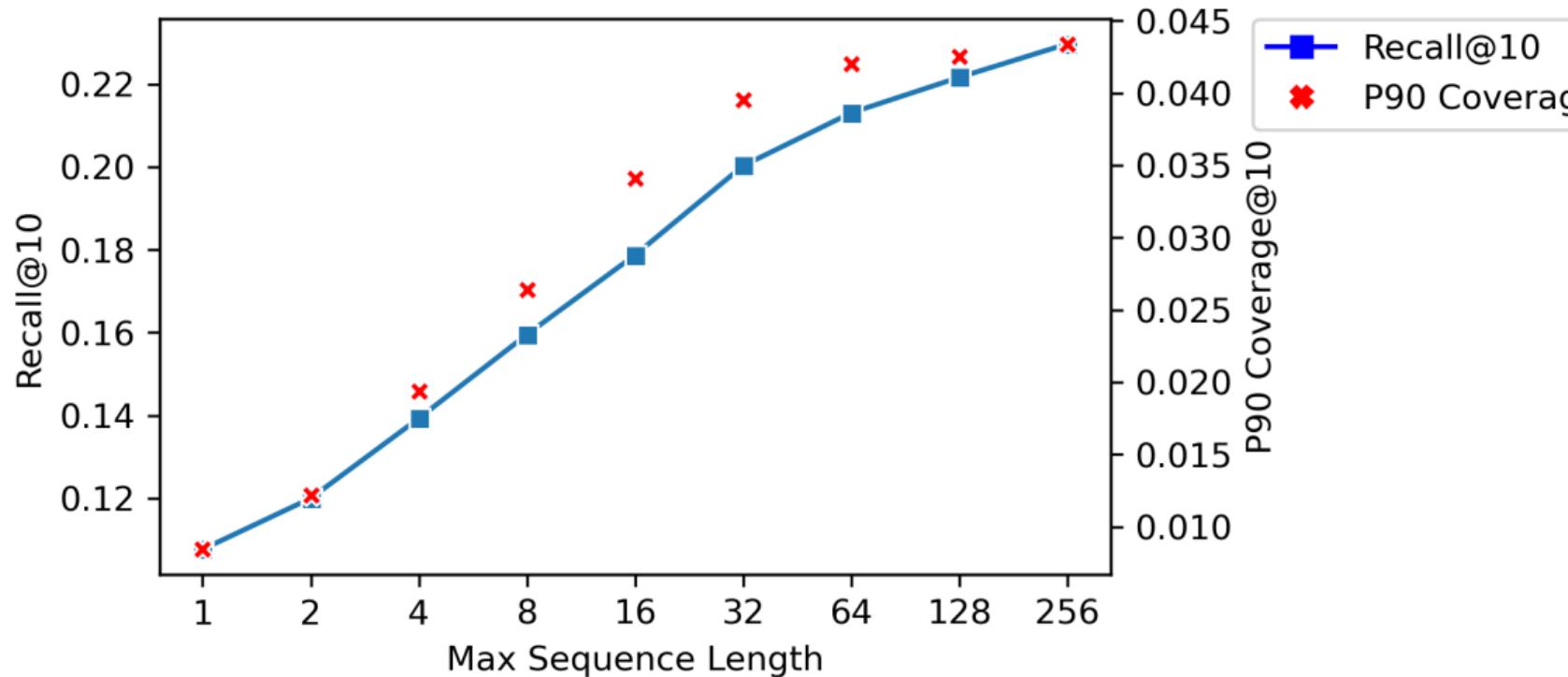


Table 7: Online A/B experiment results replacing Pinner-Sage with PINNERTFORMER as a feature in our Homefeed ranking model. We see improvements in sitewide metrics.

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%