ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

Работа с метриками и данными в рекомендательных системах

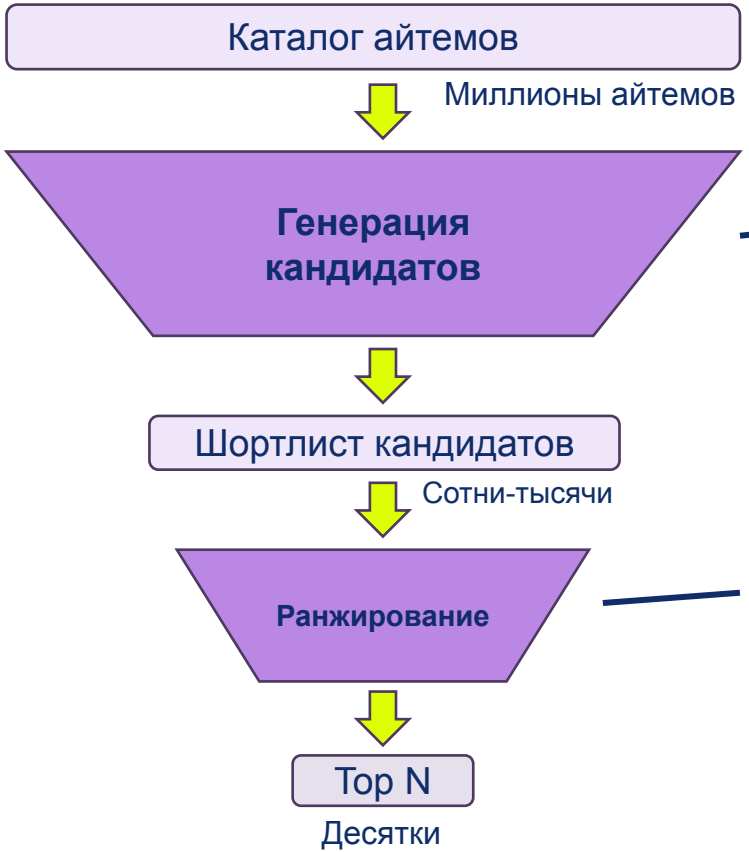
Матвеев Артем

Yandex

Яндекс

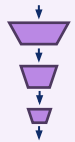
Оффлайн и онлайн метрики

Многостадийность



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

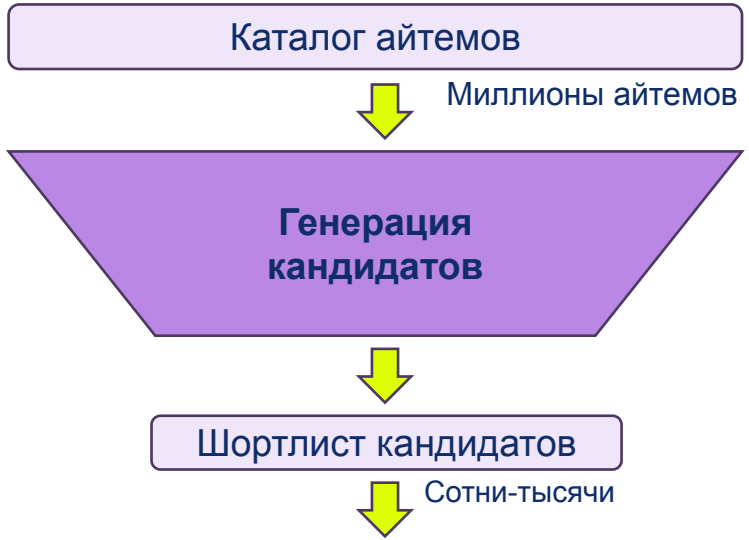
Тяжелые модели, использующие **максимум информации**

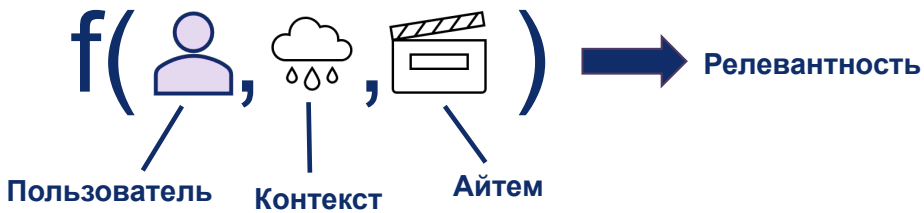


Многостадийность – это факторизация вычислений с целью ускорения.

Генерация кандидатов

- Из миллионов айтемов оставить сотни-тысячи
- Нужно уметь сравнивать на релевантность айтем со всеми остальными
- Каждый генератор выдает оценку релевантоности
- Важнее полнота, чем точность. Нужно достать все возможные релевантные, а не часть, но поднять их в высокий топ
- На практике не один генератор кандидатов, а смесь
- Консистеность ранжированию





Обозначения

- Пусть есть пользователь **u** (либо можно считать по query-ям, request-ам, выдачам, то есть добавляется еще контекст **c**). Далее под **u** будем понимать пару **(u, c)**
- В некоторый момент для него есть множество релевантных айтемов R_u
- Модель выдает ранжированный список

$$\pi_u = [i_1, \ldots, i_K]$$

- На практике для каждого объекта есть еще оценка релевантности алгоритма **a_i**, по которой происходит сортировка айтемов, ее далее опустим
- Реальная оценка релевантности айтема пользователю (был ли клик, была ли конверсия и тд)

$$\mathrm{rel}_u(i) \in \{0, 1\}$$

Генерация кандидатов: Recall@K

- Какую долю всех релевантных для юзера объектов сумели покрыть в топ-K
- K ~ 100 … 1000. Если отдаем 1000 кандидатов, то нужно и смотреть на recall@1000

$$\text{Recall@}K(u) = \frac{|R_u \cap \{i_{u,1}, \ldots, i_{u,K}\}|}{|R_u|}$$

- Агрегация по пользователям

$$\text{Recall@}K = \frac{1}{|U|} \sum_{u \in U} \text{Recall@}K(u)$$

- Нужно не забыть взять min(|R|, K)

K	Item	Rel	recall@K
1		0 ✗	0/3 = 0.000
2		1 ♥	1/2 = 0.5
3		0 ✗	1/3 = 0.333
4		1 ♥	2/3 = 0.667
5		1 ♥	3/3 = 1.000

Генерация кандидатов: HitRate@K

- Есть ли хотя бы один релевантный в топ-K
- Если у каждого **u** всего 1 релевантный, то recall = hitrate

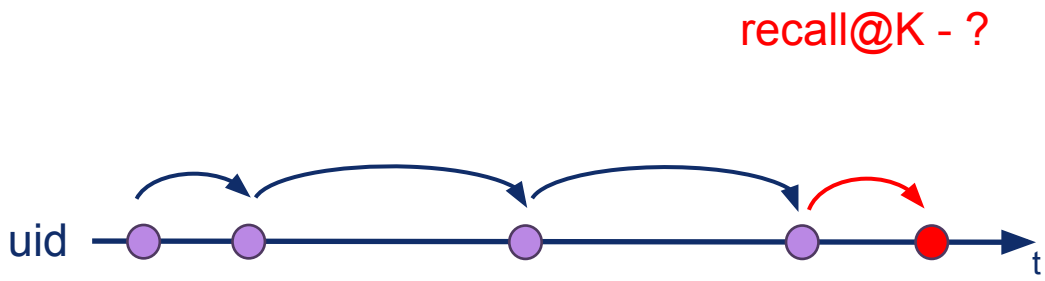
$$\mathrm{HitRate@}K(u) = \mathbb{I}\left(\left|R_u \cap \{i_{u,1}, \ldots, i_{u,K}\}\right| > 0\right)$$

$$\mathrm{HitRate@}K = \frac{1}{|U|} \sum_{u \in U} \mathrm{HitRate@}K(u)$$

K	Item	Rel	hitrate@K
1		0 	0
2		1 	1
3		0 	1
4		1 	1
5		1 	1

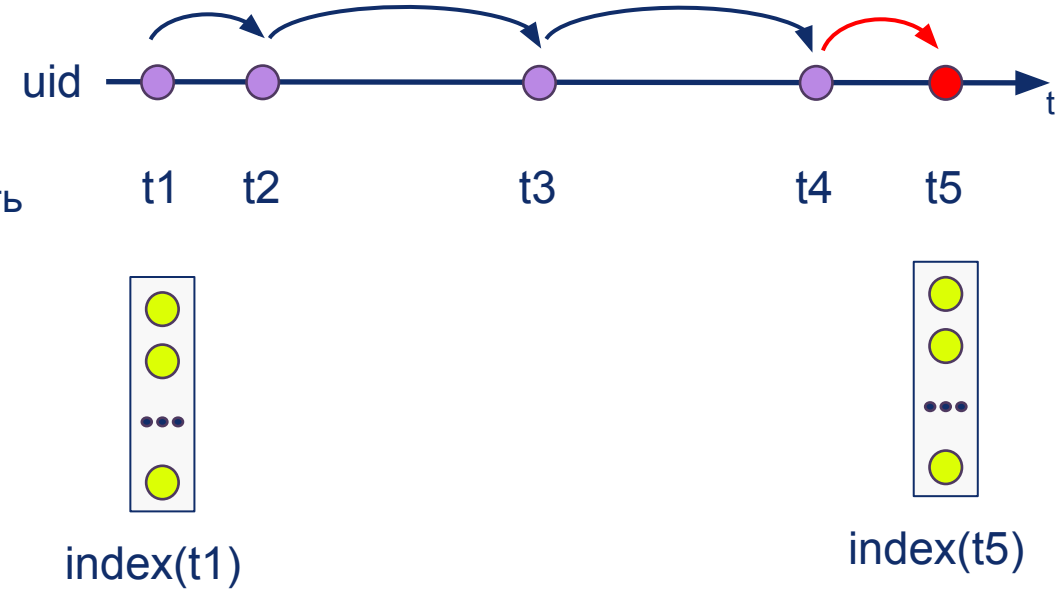
Генерация кандидатов: Next Item Prediction

- Некоторой моделью решаем задачу NIP
- Хотим замерить качество нашей модели - recall@K
- Откуда брать множество айтемов, среди которого отбираем топ-K?



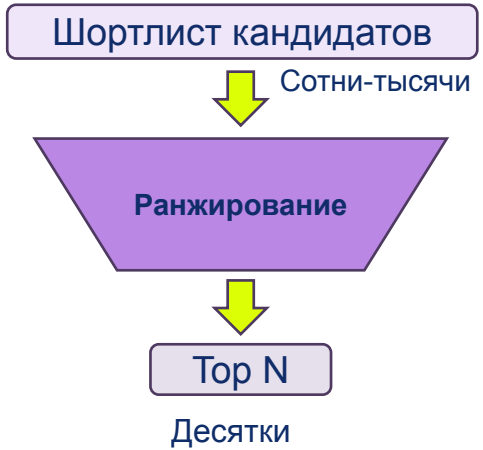
Генерация кандидатов: Негативы из индекса

- В реальных системах айтемы исчезают и появляются со временем
- **index(t)** - актуальный набор айтемов в системе на момент времени t
- При замере метрик важно брать наиболее актуальный индекс



Ранжирование

- Модель отдает оценку на то, будет ли положительных фидбек пользователя с айтемом
- Нужно уметь сравнивать на релевантность хороший айтем с другими хорошими (**важно отличие от генерации кандидатов, задача усложняется**)
- Важнее точность, чем полнота. Важно, чтобы релевантный айтем был в топе, т.к. отдаем топ 10 пользователю



Ранжирование: NDCG@K

- Учитываем позиции, попадание на 1 место важнее, чем 10
- K ~ 10 (итоговая выдача)
- Discounted Cumulative Gain

$$\mathrm{DCG@}K(u) = \sum_{j=1}^{K} \frac{2^{\mathrm{rel}_u(i_{u,j})} - 1}{\log_2(j+1)}$$

- IDCG - DCG идеального ранжирования (релевантные вверху)
- Normalized Discounted Cumulative Gain

$$\mathrm{NDCG@}K(u) = \frac{\mathrm{DCG@}K(u)}{\mathrm{IDCG@}K(u)}$$

$$\mathrm{NDCG@}K = \frac{1}{|U|} \sum_{u \in U} \mathrm{NDCG@}K(u)$$

k	item	rel	discount	DCG@k	IDCG@k	NDCG@k
1	a 	0 	1.000	0.000	1.000	0.000
2	b 	1 	0.631	0.631	1.631	0.387
3	c 	0 	0.500	0.631	2.131	0.296
4	d 	1 	0.431	1.062	2.131	0.498
5	e 	1 	0.387	1.449	2.131	0.680

Ранжирование: MAP@K

- Насколько рано встречаются релевантные; награждает за ранние корректные позиции
- Precison@j

$$\mathrm{P@}j(u) = \frac{1}{j}\sum_{t=1}^{j}\mathbb{I}(\mathrm{rel}_u(i_{u,t}) > 0)$$

- Average Precison@K

$$\mathrm{AP@}K(u) = \frac{1}{\min(|R_u|, K)}\sum_{j=1}^{K}\mathrm{P@}j(u)\cdot\mathbb{I}(\mathrm{rel}_u(i_{u,j}) > 0)$$

- MAP@K

$$\mathrm{MAP@}K = \frac{1}{|U|}\sum_{u\in U}\mathrm{AP@}K(u)$$

k	item	rel	P@k	P@k * rel	AP@k
1	a 	0 	0.000	0.000	0.000
2	b 	1 	0.500	0.500	0.250
3	c 	0 	0.333	0.000	0.167
4	d 	1 	0.500	0.500	0.333
5	e 	1 	0.600	0.600	0.533

Ранжирование: AUC-ROC

- Пользователь видит только ранжированный топ-K
- K ~ 10
- AUC = вероятность правильно отражировать пару P(y+ > y-)
- По-другому еще QueryAUC

Ranking type

$$\frac{\sum_q \sum_{i,j \in q} \sum I(a_i, a_j) \cdot w_i \cdot w_j}{\sum_q \sum_{i,j \in q} \sum w_i * w_j}$$

The sum is calculated on all pairs of objects (i, j) such that:

- $t_i < t_j$
- $I(x, y) = \begin{cases} 0, & x < y \\ 0.5, & x = y \\ 1, & x > y \end{cases}$

https://catboost.ai/docs/en/concepts/loss-functions-ranking#ranking-type1

Ранжирование: AUC-ROC

4 пары позитив против негатива:

b > a, 2 > 1 => 0

b > c, 2 < 3 => 1

b > d, 2 < 4 => 1

b > e, 2 < 5 => 1

QueryAUC = ¾ = 0.75

K	Item	Rel
1	a 	0 
2	b 	1 
3	c 	0 
4	d 	0 
5	e 	0 

Ранжирование: MRR (Mean Reciprocal Rank)

- На какой позиции стоит первый релевантный

$$r_u = \min\{j : \ \mathrm{rel}_u(i_{u,j}) > 0\}$$

- Если нет релевантных — считаем вклад 0

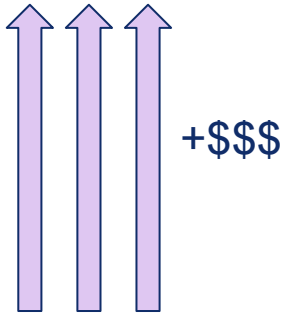
$$\mathrm{RR}(u) = \frac{1}{r_u}, \qquad \mathrm{MRR} = \frac{1}{|U|}\sum_{u \in U} \mathrm{RR}(u)$$

$$\mathrm{MRR@}K = \frac{1}{|U|}\sum_{u \in U} \mathbb{I}(r_u \le K)\frac{1}{r_u}$$

K	Item	Rel	MRR@K
1		0 	0
2		1 	0.5
3		0 	0.5
4		1 	0.5
5		1 	0.5

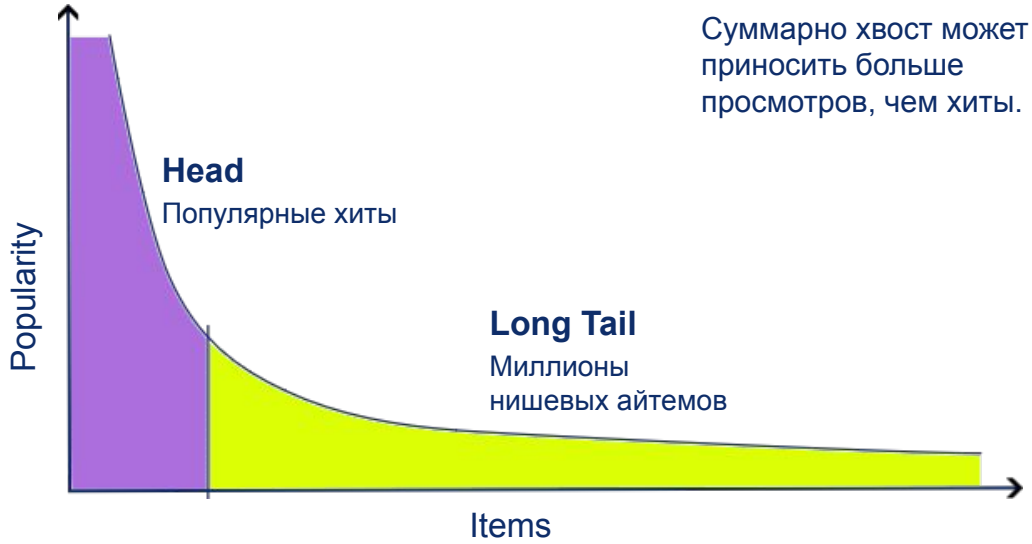
Ранжирование: попугаи

- Замеряем качество алгоритма ранжирования через NDCG@10
- Старый имеет качество **NDCG_OLD**
- Новый имеет качество **NDCG_NEW**
- **ППГ = (NDCG_NEW - NDCG_OLD) / NDCG_OLD * 1000**
- То есть измеряем относительные приросты в тысячах ппг-ев
- 2000 ппг - круто! Видимо в АБ



Тяжелый хвост

- Важно делать распределение менее скошенным
- Тяжелый хвост = айтемы, с которыми было меньше threshold взаимодействий
- Можем замерять долю тяжелого хвоста в выдаче



Тяжелый хвост: Coverage@K

- Как хорошо мы покрываем наш индекс (доля показанных айтемов во всем каталоге)
- Пусть **I** — все айтемы в каталоге.

$$\mathrm{ItemCoverage@}K = \frac{\left|\bigcup_{u \in U} \{i_{u,1}, \ldots, i_{u,K}\}\right|}{|I|}$$

Холодный срез

- Каталоги постоянно обновляется
- Для новых айтемов отсутствует пользовательских фидбек
- Измеряем те же метрики, но на срезе холодных айтемов (ground truth = холодный айтем)
- Можно аналогично замерять на срезе холодных пользователей
- Холодный айтем = недавно добавлен в систему без взаимодействий
- Холодный пользователь = пользователь, у которого мало взаимодействий в системе

Distribution drift

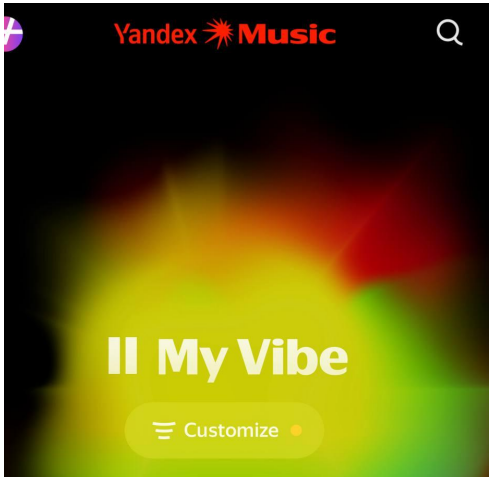
- Очень быстро меняются интересы пользователей, тренды и каталоги айтемов
- Модели со временем деградируют
- В отличие от LLM модели постоянно дообучаются на новых данных
- Можем измерять, как быстро деградирует качество

Онлайн метрики

- Модель подбирается по оффлайн метрикам
- Проводится АБ тест против старой версии рекомендательной системы
- Хотим корреляцию оффлайн и онлайн метрик
- CTR, Likes, Cart, TLT, CVR, Diversity, GMV, Session Length, Skip Rate, Exploration

Работа с данными

Рекомендательные логи: музыка



- Взаимодействия пользователя
- Триллионы взаимодействий
- Информация про пользователя, контекст, айтем, фидбек

uid	timestamp	device	is_organic	item_id	…	listen time	like	dislike
123	193031093	iphone	True	52	…	10s	1	0
321	193031099	android	False	13	…	150s	0	1
…	…	…	…	…	…	…	…	…
999	194031099	mac	False	572	…	76s	0	0

Yambda-5B - A Large-Scale Multi-modal Dataset for Ranking And Retrieval

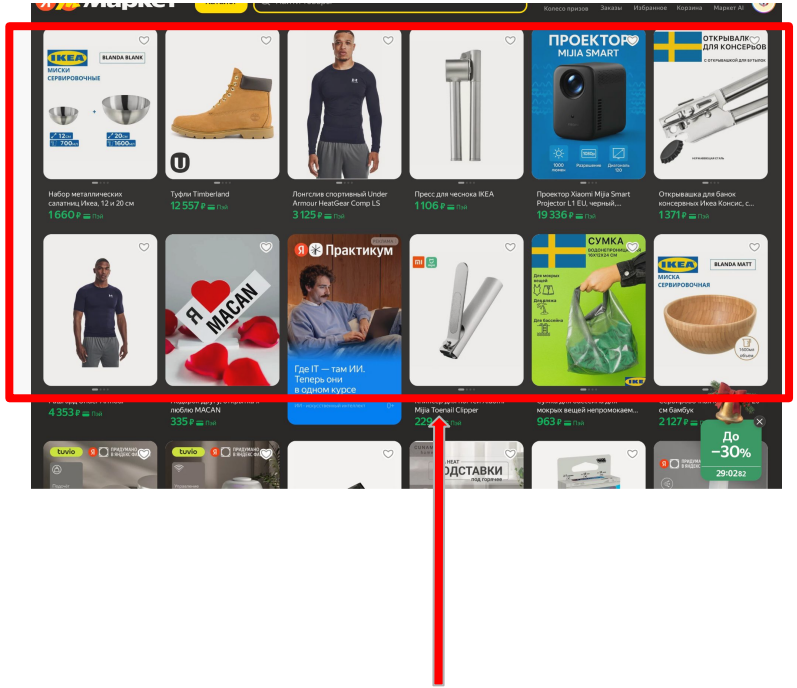
Рекомендательные логи: маркет

- impression = показ, набор айтемов, которые были показаны пользователю

uid	timestamp	device	impression

123	193031093	iphone	{
			item_1: click,
			item_2: view,
			item_3: view,
			item_4: cart (!)
			}

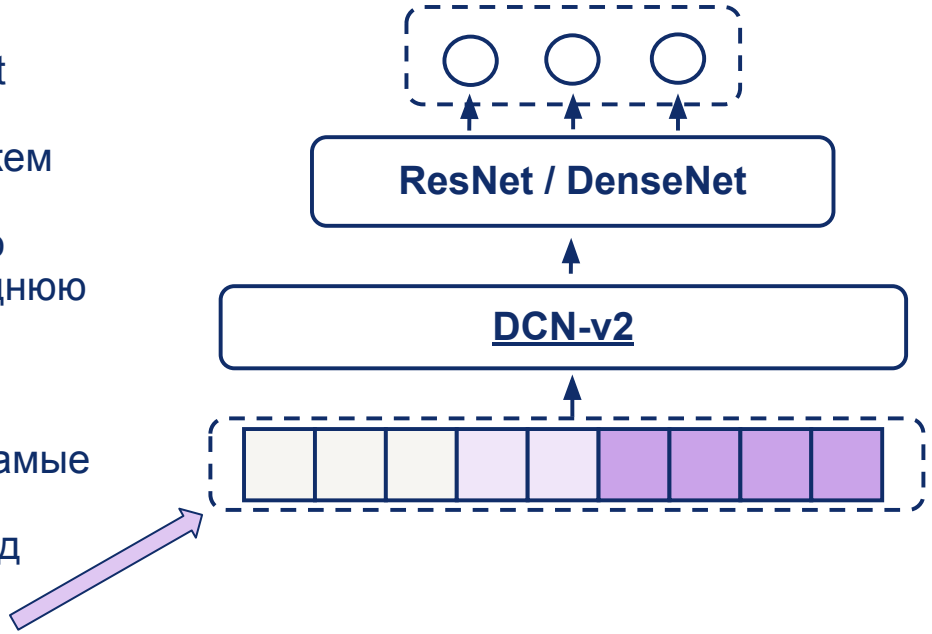
321	193031099	android	{
			item_11: view,
			item_22: view,
			item_33: view,
			item_44: view
			}



impression

Счетчики

- Подаем в нейросетевую модель/catboost информацию про пользователя,
- На основе рекомендательных логов можем насчитать “счетчики”
- По пользователю: число лайков, сколько всего взаимодействий, лайков за последнюю неделю, любимый артист и тд
- По айтему: сколько раз его лайкнули, популярность, артист и тд
- Кросс-счетчики пользователь x айтем (самые сильные): сколько раз пользователь прослушал этот айтем, этого артиста и тд



счетчики

Обучающий семпл для модели со счетчиками

- Чаще всего ранжирование
- Счетчиков 100-1000
- Таргетов может быть несколько: лайк, доля прослушивания, пара (item_1, item_2) правильно отранжированна

uid	cnt1	cnt2	…	cnt999	target1	target2
123	424	0.314	…	1000	0	0.42
…	…	…	…	…	…	…
321	524	0.594	…	424	1	0.91

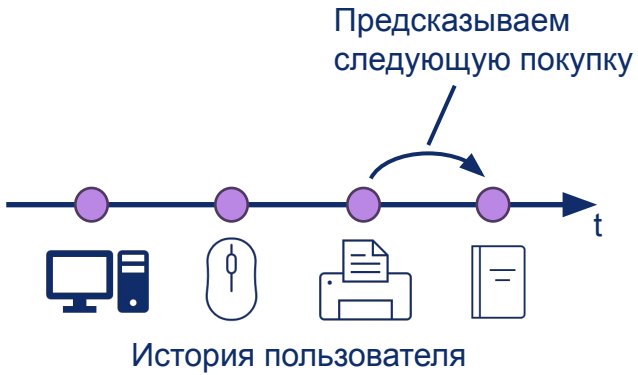
Редьюс истории пользователя

- Можем обрабатывать не счетчики, а сразу сырые истории
- Сортируем по uid, timestamp
- Агрегируем в единый список

uid ⇩	timestamp ⇩	device	is_organic	item_id	…	listen time	like	dislike
123	193031093	iphone	True	52	…	10s	1	0
123	193031099	iphone	False	13	…	150s	0	1
…	…	…	…	…	…	…	…	…
123	194031099	iphone	False	572	…	76s	0	0

Обучающий пример для модели с историями

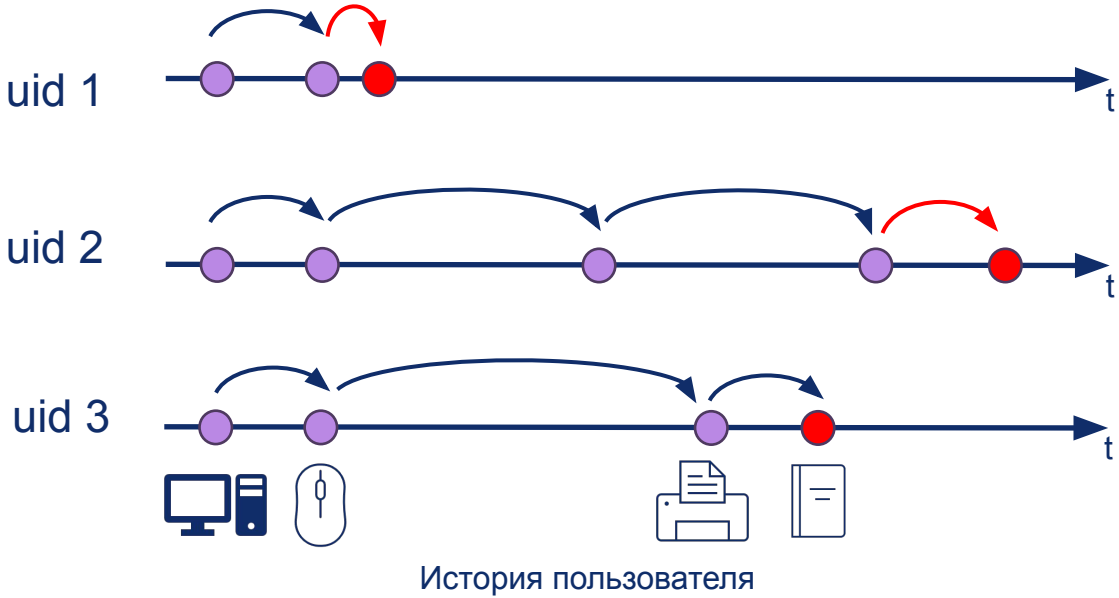
- Например, трансформер над историей пользователя



uid	timestamp ⇧	is_organic	item_id	target (item_id)
123	[193031093, 193031095,..., 194038091]	[True, False, …, True]	[52, 12, …, 49]	42
…	…	…	…	

Оценка качества моделей: leave-one-out

- Считаем recall@K по самому последнему взаимодействию
- В качестве негативов берем индекс всех айтемов, ground truth = следующий айтем
- Ликуем будущее при обучении, популярность айтемов, паттерны из будущего



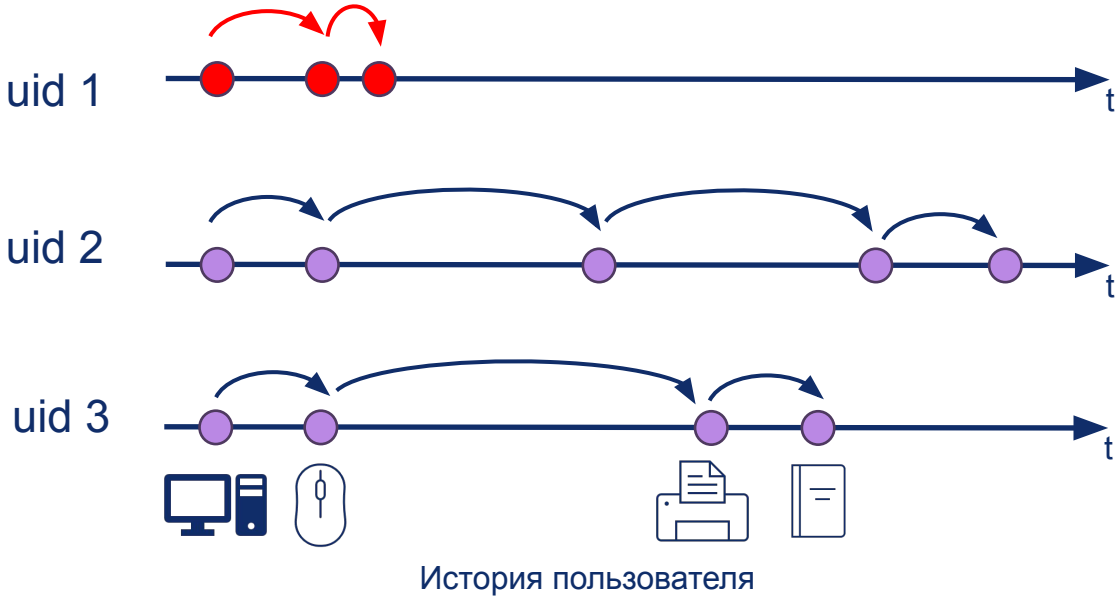
История пользователя

train

eval

Оценка качества моделей: user-split

- Часть пользователей в train, часть в eval
- Тот же лик



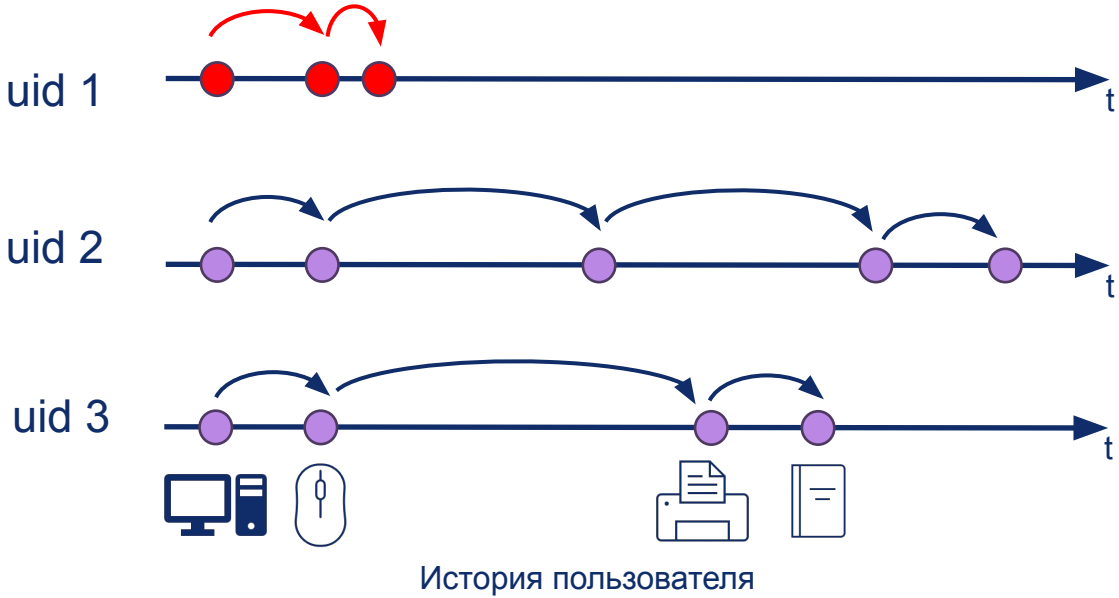
История пользователя

train

eval

Оценка качества моделей: user-split

- Часть пользователей в train, часть в eval
- Тот же лик



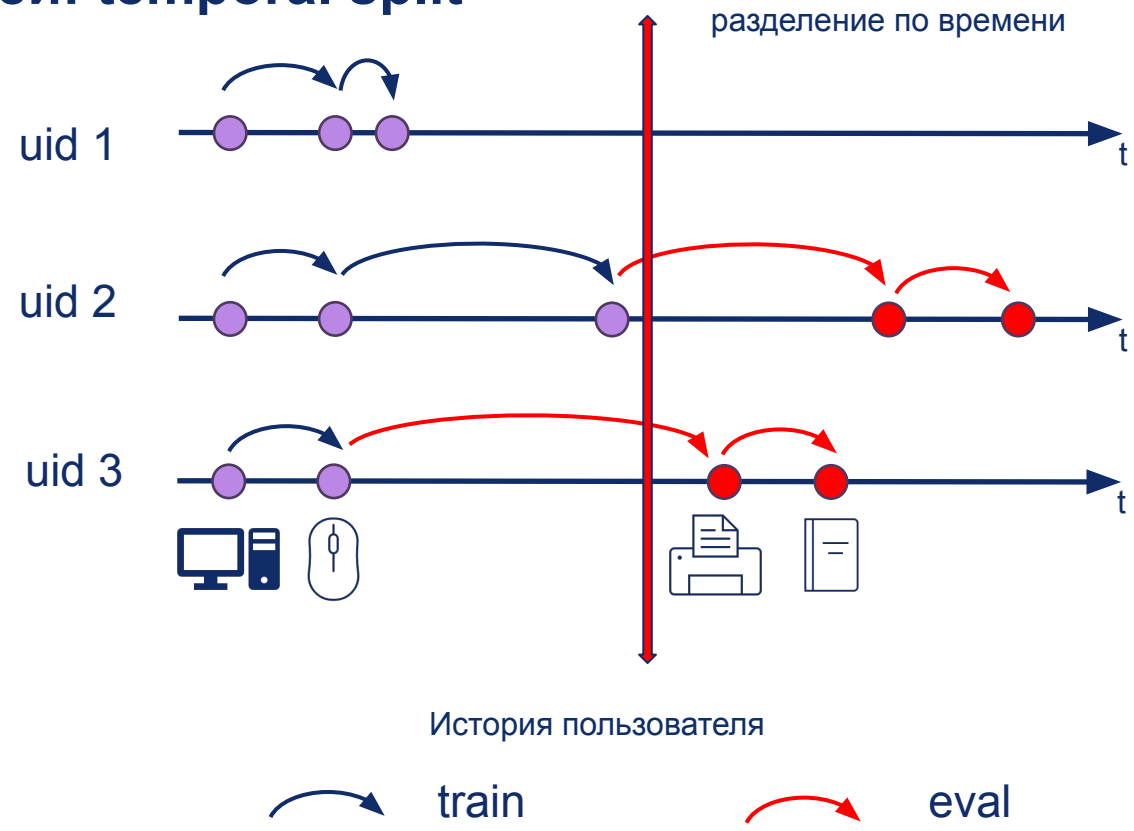
История пользователя

train

eval

Оценка качества моделей: temporal split

- Так надо!
- Выровнено с тем, как реально выкатываются новые алгоритмы



Посмотрим на код