

# ML дизайн рексистем

Нейросетевые рекомендательные  
системы 2025/26, лекция 2

Кирилл Хрыльченко

Старший преподаватель,  
ФКН ВШЭ

Яндекс

# План занятия

На лекции:

1. Метрики
2. Данные и логи
3. Генерация кандидатов
4. Ранжирование и лоссы
5. Бонусная секция: про мой R&D опыт в Яндексе

На семинаре:

1. Классические бейзлайны
2. Разбор статьи про датасет Yambda

# Вы оказались на необитаемом острове

Вам нужно построить рекомендательную систему с нуля.

С чего начать?

# Метрики

Часть 1

# Всё начинается с метрик

Почему:

- У платформы всегда есть цель, с которой она делает рексистему
- Данные, модели и алгоритмы – это всего лишь инструменты
- Неправильная метрика – плохая рексистема

# Какие цели у вашей рексистемы?

На примере YouTube:

- Для рекламы хотим большой **total watch time**
- Чтобырастить пользовательскую базу, хотим большой **retention** (и **satisfaction** пользователей)
- Чтобырастить авторскую базу, нужно растить **long-tail exposure** (рекомендовать нишевый контент)

В реальности – хотим всё и сразу.

# А клики?

- В рекомендациях почти никогда не хотим оптимизировать (только) клики
- Оптимизация кликов приводит к росту **кликбейта**
- Один из частых красных флагов на собеседованиях – кандидат предлагает оптимизировать клики
- Где нужно оптимизировать клики – в рекламе

Что оптимизируем – то и получим.

# Retention

- Все рекомендательные системы хотят большой retention

# Retention

- Все рекомендательные системы хотят большой **retention** (и tinder тоже)
- Его сложно замерять и ещё более тяжело оптимизировать. Почему?

# Retention

- Все рекомендательные системы хотят большой retention (и tinder тоже)
- Его сложно замерять и ещё более тяжело оптимизировать. Почему?
  - Чтобы замерить 6-недельный retention, нужно ждать 6 недель :)
  - Вопрос: какой длительности обычно A/B тесты в рекомендательных системах?
- Что можем сделать:
  - Считать краткосрочный retention – e.g., 1-day retention, 7-day retention; количество дней активности

# Прокси-метрики

Подбираем метрику, которая быстро считается и коррелирует с целевой.

На примере retention:

- Берём данные за большой горизонт
- Считаем для них настоящий retention
- Подбираем комбинацию всевозможных краткосрочных метрик (времени просмотра, количества лайков и тд)
  - В первую неделю данных
  - Коррелирующую с retention (а как?)

# Прокси-метрики

Подбираем метрику, которая быстро считается и коррелирует с целевой.

На примере retention:

- Берём данные за большой горизонт
- Считаем для них настоящий retention
- Подбираем комбинацию всевозможных краткосрочных метрик (времени просмотра, количества лайков и тд)
  - В первую неделю данных
  - Коррелирующую с retention (а как? Линейная регрессия)

# Выводы

- Рексистемы начинаются с метрик
- Неправильная метрика – плохая рексистема
- Важных метрик обычно несколько
- Клики – почти всегда плохая идея
- Долгосрочные метрики сложно замерять и оптимизировать

# Данные

Часть 2

# Данные

- Наши возможности сильно зависят от данных
- Большинство улучшений рексистемы связаны с данными
- Хотим логировать всё, что движется

# Логи

Логи – это главный источник золота:

- Это таблицы с событиями
- Обычно разбиты по дням (иногда по часам)
- Каждая строка – одно событие
- Все, что происходит в системе – превращается в события
- Логи с кликами, с заказами, лайками, etc

Речь идёт именно про рекомендательные логи.

uid	timestamp	item_id	is_organic	playe
uint32	uint32	uint32	uint8	uint1
10	3,848,010	6,490,652	0	0
10	3,848,650	686,823	0	0
10	3,853,270	8,761,089	0	0
10	3,853,395	9,171,216	0	0
10	3,854,490	6,008,476	0	0
10	3,855,065	8,796,865	0	0
10	3,855,530	2,971,017	0	0
10	3,855,680	1,524,606	0	0
10	3,855,735	1,524,606	0	0
10	5,648,185	7,053,616		1

<https://huggingface.co/datasets/yandex/yambda>

# События

Что знаем про события:

- Время (timestamp)
- Идентификаторы:
  - e.g., User ID, Item ID, Request ID
- Контекст:
  - Устройство
  - Клиент (web / mobile)
  - Поверхность (моя волна, плейлист дня, мои лайки)

# Метаданные

Например, у айтемов есть:

- Названия
- Описания
- Категории
- Изображения

ItemID	Title	Description	Categories	Brand	Semantic ID

[Recommender Systems with Generative Retrieval](#)

Эти данные “тяжелые”, редко меняются, чаще всего не хотим их дублировать и храним отдельно.

# Зачем нам логи

- Обучение моделей
- Оценка качества
- Строим рекомендации (оффлайн-модели) на логах
- Что мы можем – зависит напрямую от того, что есть в логах

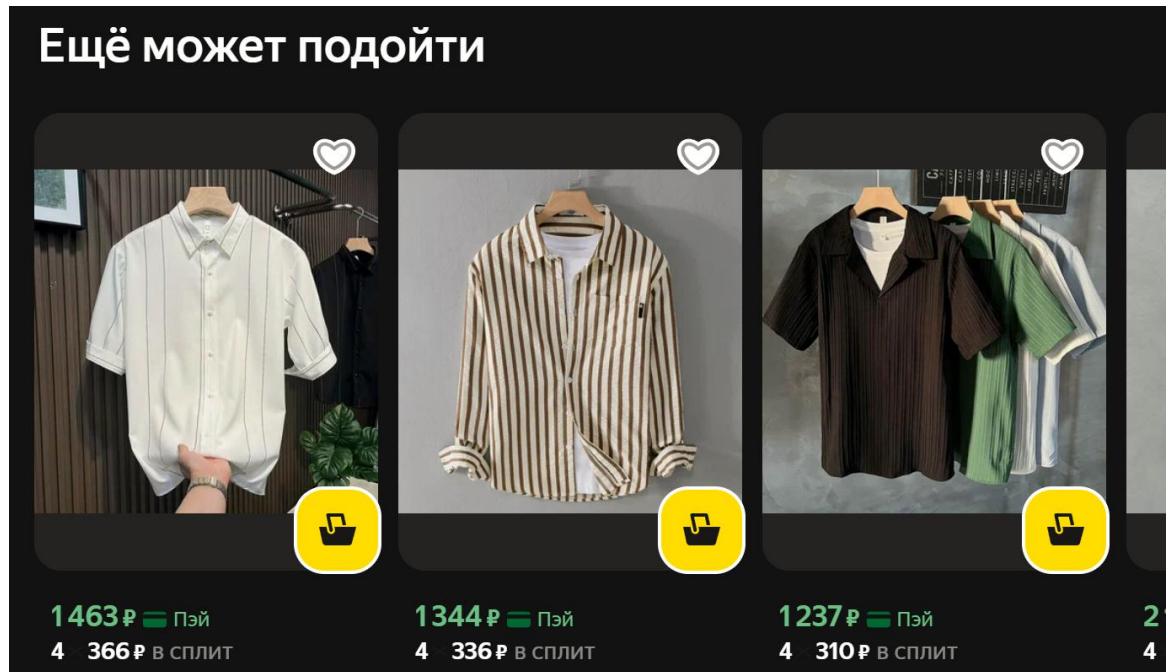
На логах держится буквально вся рекомендательная система.

# Impressions

Impressions – это особый лог:

- Все, что показали пользователю в рекомендациях
- Независимо от (наличия) реакции
- Это **самый** большой по объёму лог

Q: Как понять какие показы рекомендаций пользователь проигнорировал?



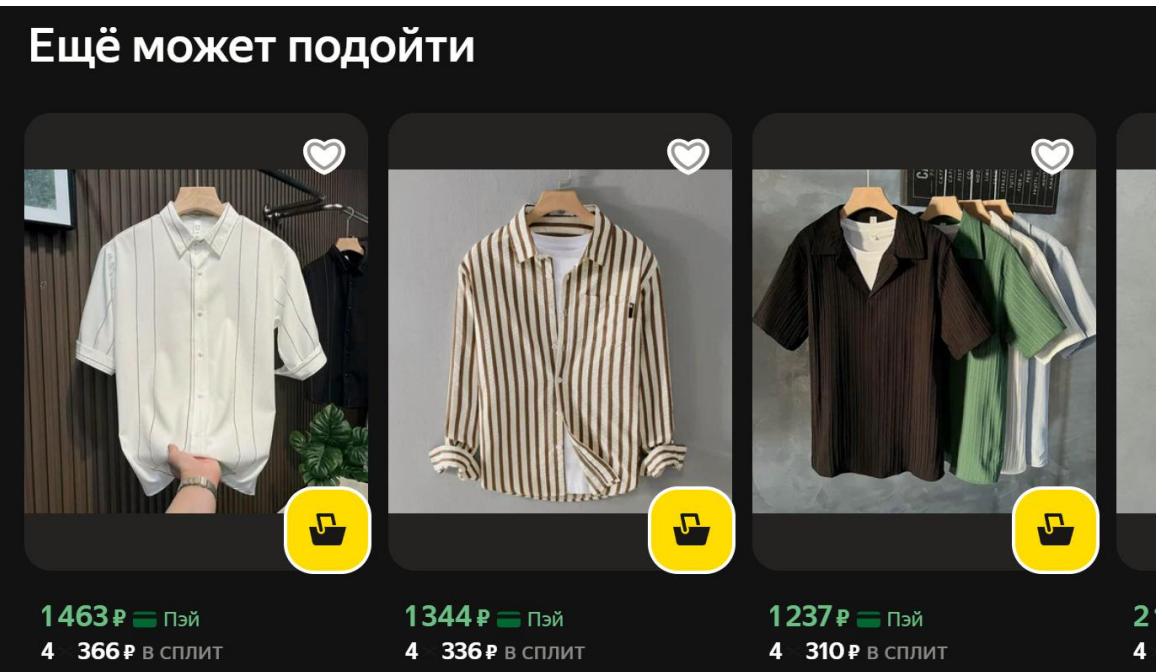
# Impressions

Impressions – это особый лог:

- Все, что показали пользователю в рекомендациях
- Независимо от (наличия) реакции
- Это **самый большой по объёму лог**

Q: Как понять какие показы рекомендаций пользователь проигнорировал?

A: Сделать джойн с другими событийными логами



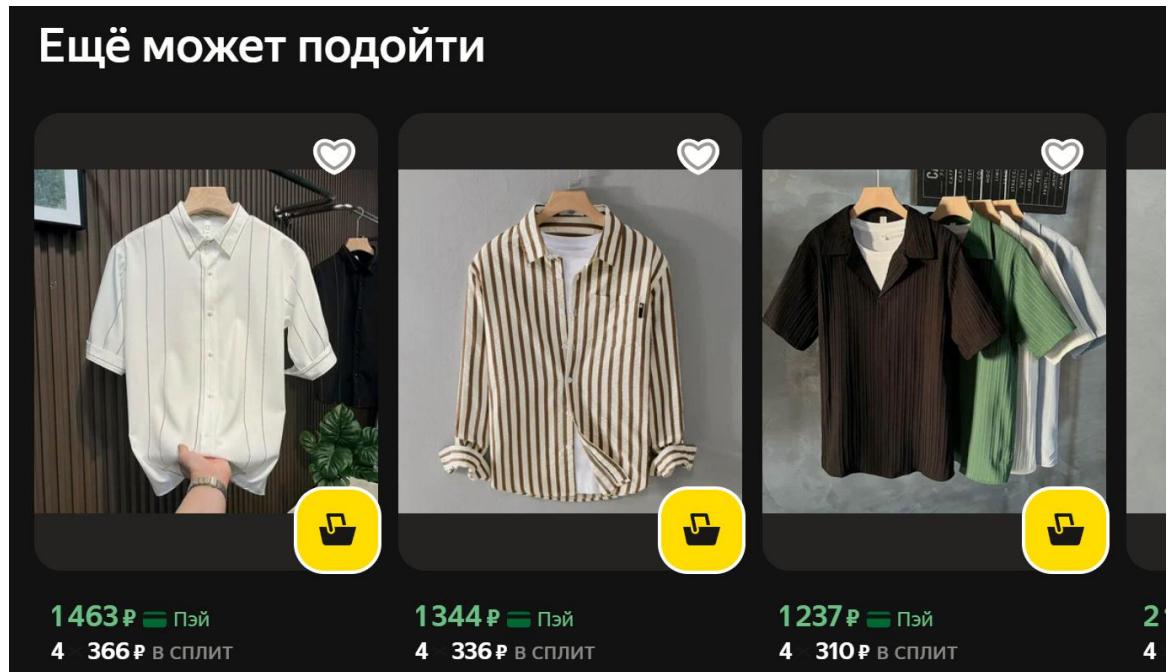
# Impressions

Impressions – это особый лог:

- Все, что показали пользователю в рекомендациях
- Независимо от (наличия) реакции
- Это **самый большой по объёму лог**

Q: Как понять какие показы рекомендаций пользователь проигнорировал?

A: Сделать джойн с другими событийными логами с помощью (Request ID, Item ID)



# Отложенный фидбек

Иногда результат рекомендации прорастает в систему очень поздно:

- Заказы происходят спустя дни после показа
- Возникает проблема атрибуции
- Используем lookahead – “приклеиваем” к показам заказы в нужном горизонте по (user ID, item ID)



# Выводы

- Данные определяют наши возможности
- Impressions – особый лог
  - он нужен как минимум для обучения моделей
- Приклеивать фидбек к impressions – не всегда тривиально
  - особенно отложенный фидбек

# Генерация кандидатов

Часть 3

# Многостадийность

Рекомендации – многостадийный процесс:

- На ранних стадиях – более быстрые и эффективные алгоритмы
- Дальше – более тяжелые модели

Если размер каталога айтемов небольшой, то можно обойтись одной стадией.

Скорим всё: как мы упростили ML-стек рекомендаций в Лавке и отказались от кандидатогенерации (Марк Нарусов)

# Генерация кандидатов

Задача генерации кандидатов:

- Сформировать шортлист кандидатов
- Не упустить хороших кандидатов (обеспечить полноту)
- Дать разные опции (обеспечить разнообразие)
- Сделать это очень быстро

# Источники кандидатов

Примеры простых кандидатов:

- История пользователя
- Топ популярного
- Item-to-item
- Близости по эмбеддингам, построенным любым образом

Больше на сегодняшнем семинаре

# Пример: item-to-item

- Формируем для каждого айтема списки похожих айтемов
- Берём последние действия пользователя (айтемы)
- Объединяем списки для этих айтемов

Item2Item: классика жанра

	$item_1$	$item_2$	$item_3$	$item_4$	$item_5$	$item_6$	$item_7$	$item_m$
$user_1$	0	1	1	0	0	1	1	0
$user_2$	1	1	0	1	1	1	0	0
$user_3$	0	1	1	0	0	0	1	1
$user_4$	0	0	0	1	1	1	0	1
$user_5$	0	0	1	0	1	0	1	1
$user_6$	0	1	1	1	1	1	0	1
$user_7$	1	1	0	1	0	1	1	0
$user_8$	0	0	1	0	1	1	1	0
$user_9$	0	1	0	0	1	1	0	0

$$s(i, j) = \frac{\sum_{u \in U_i \cap U_j} r_{ui} r_{uj}}{\sqrt{\sum_{u \in U_i} r_{ui}^2} \sqrt{\sum_{u \in U_j} r_{uj}^2}}$$

1. Похожесть между айтемами – cosine similarity:

2. В оффлайне считаем список похожих для каждого айтема

3. Достаем по взаимодействиям пользователя похожие айтемы и ранжируем их:

$$\hat{r}_{ui} = \frac{\sum_{j \in I_u} s(i, j) r_{uj}}{\sum_{j \in I_u} |s(i, j)|}$$

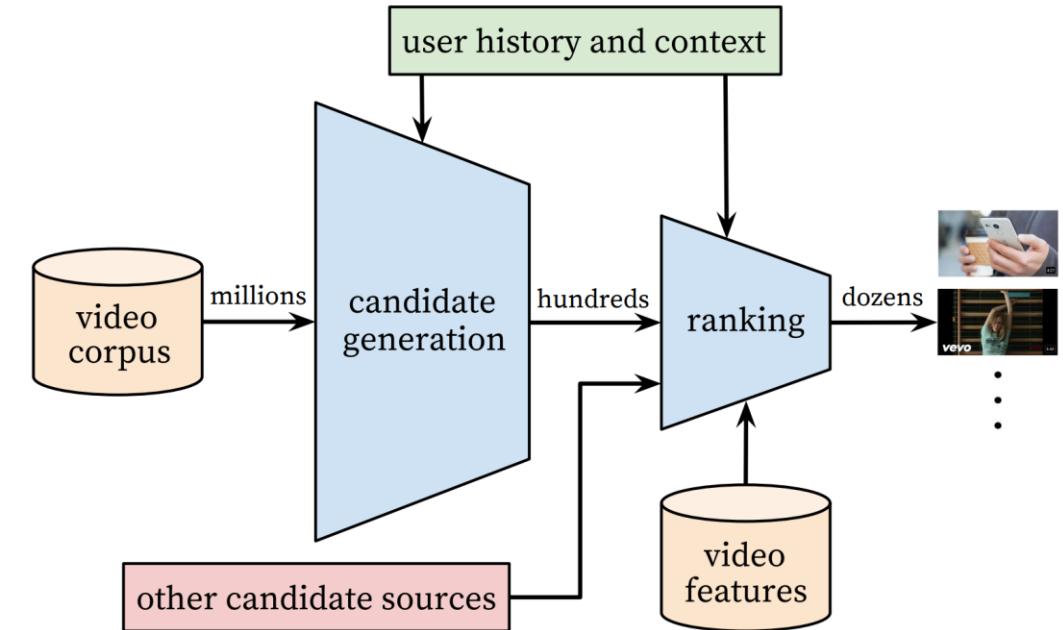
Переосмысление item2item-рекомендаций в Дзене (Андрей Зимовнов, 2024)  
Переосмысление рекомендаций в Дзене и внедрение item2item-схемы (Дмитрий Шишов, 2023)

# Вопросы

- Q: Сколько кандидатов возвращаем из кандгена?
- Q: Какой кандген из предложенного списка будем использовать?

# Вопросы

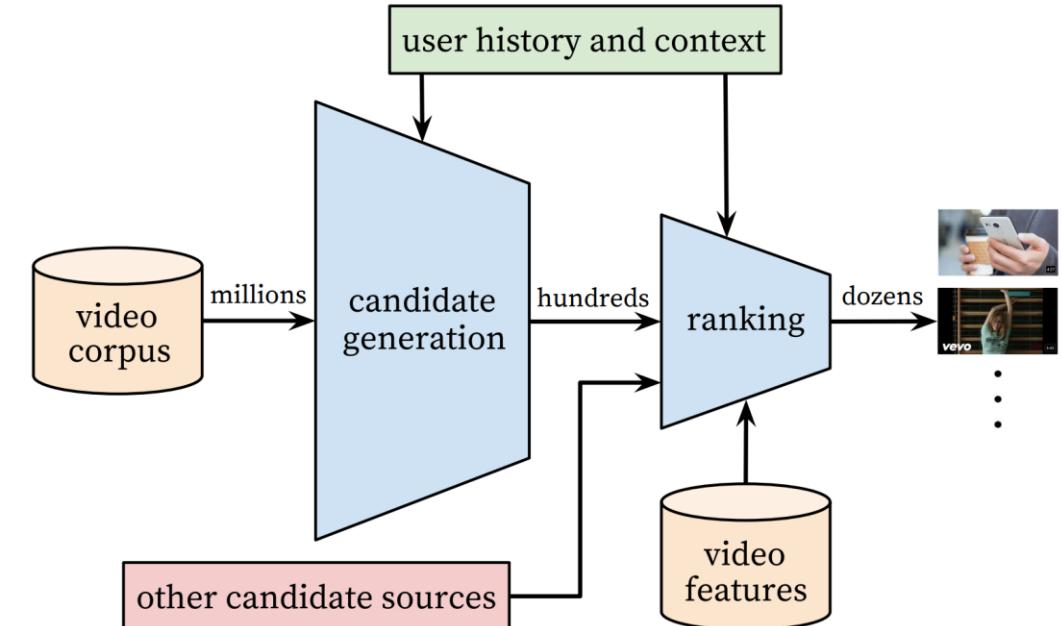
- **Q: Сколько кандидатов возвращаем из кандгена?**
  - А: От сотен до десятков тысяч
- **Q: Какой кандген из предложенного списка будем использовать?**
  - А: Все



[Deep Neural Networks for YouTube Recommendations](#)

# Вопросы

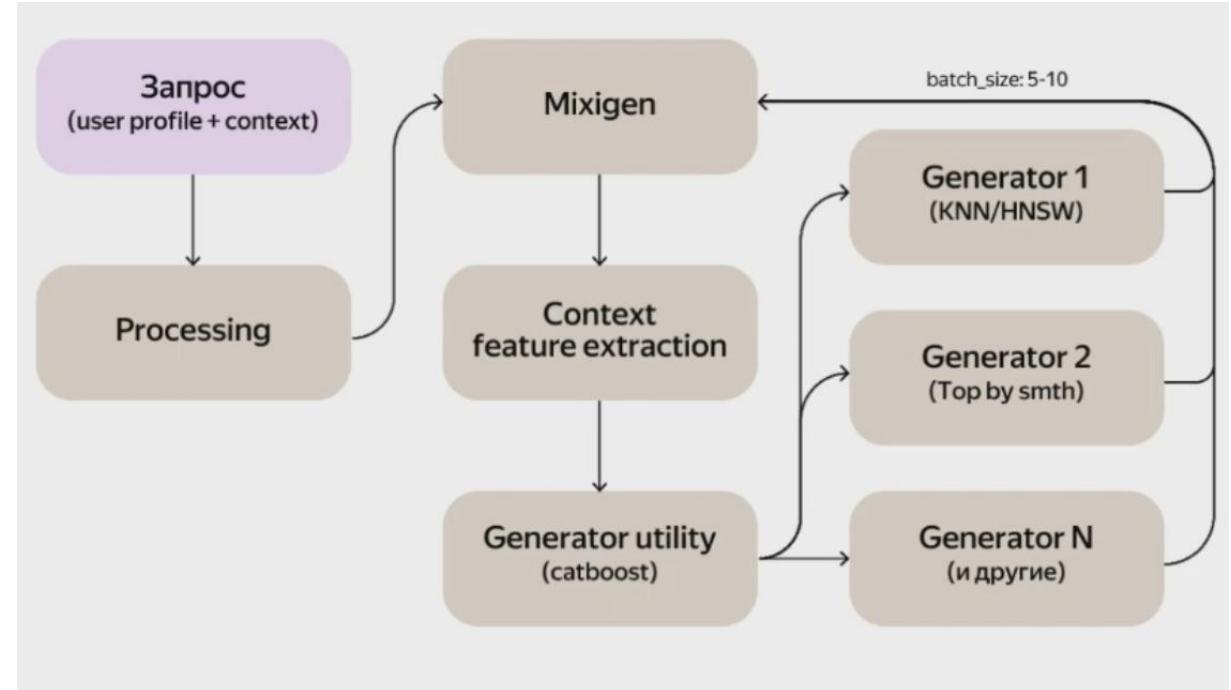
- **Q: Сколько кандидатов возвращаем из кандгена?**
  - А: От сотен до десятков тысяч
- **Q: Какой кандген из предложенного списка будем использовать?**
  - А: Все
- **Q: Как объединить несколько источников кандидатов?**



[Deep Neural Networks for YouTube Recommendations](#)

# Объединение источников кандидатов

- Брать поровну – неоптимально
- Можно сделать алгоритм, адаптивно определяющий квоту для каждого источника (e.g., Mixigen)



Про Mixigen:

<https://t.me/WazowskiRecommends/58>

<https://t.me/WazowskiRecommends/59>

<https://t.me/WazowskiRecommends/61>

Архитектура бесконечной персональной ленты  
Яндекс Маркета (Владимир Бондаревский)

Бонусом: Адаптивное квотирование источников  
кандидатов в near real-time (Даниил Самойлов, VK)

# Объединение источников кандидатов

Более простой вариант:

- набрать кандидатов с запасом
- добавить стадию преранжирования



[Тренировки по ML 2.0 Лекция 7:](#)  
[Рекомендательные системы \(Савва Степурин, 2024\)](#)

# Оценка качества источника кандидатов

На семинаре рассказывали про метрику Recall@K:

- Представим что ваш кандген уже работает в проде и выдает 100 кандидатов
- Он работает в проде, фидбек логируется
- Хотим посчитать для модели Recall@100.
- Какой он будет?

# Оценка качества источника кандидатов

На семинаре рассказывали про метрику Recall@K:

- Представим что ваш кандген уже работает в проде и выдает 100 кандидатов
- Он работает в проде, фидбек логируется
- Хотим посчитать для модели Recall@100.
- Какой он будет? 100%
- Что с этим делать:
  - Органический трафик
  - Несколько источников кандгена
  - Смотрим на Recall@K с небольшим K
  - Страдать

# Неразнообразные рекомендации

Пользователи жалуются, что в рекомендациях мало разнообразия. Что делать?

# Неразнообразные рекомендации

Пользователи жалуются, что в рекомендациях мало разнообразия. Что делать?

- В первую очередь нужно смотреть на разнообразие кандидатов
- Если кандидаты не обеспечивают нужные свойства системы, то улучшение ранжирования не поможет
- Метрика Coverage@K (см. семинар)

А ещё – вводить стадию переранжирования с учётом разнообразия.

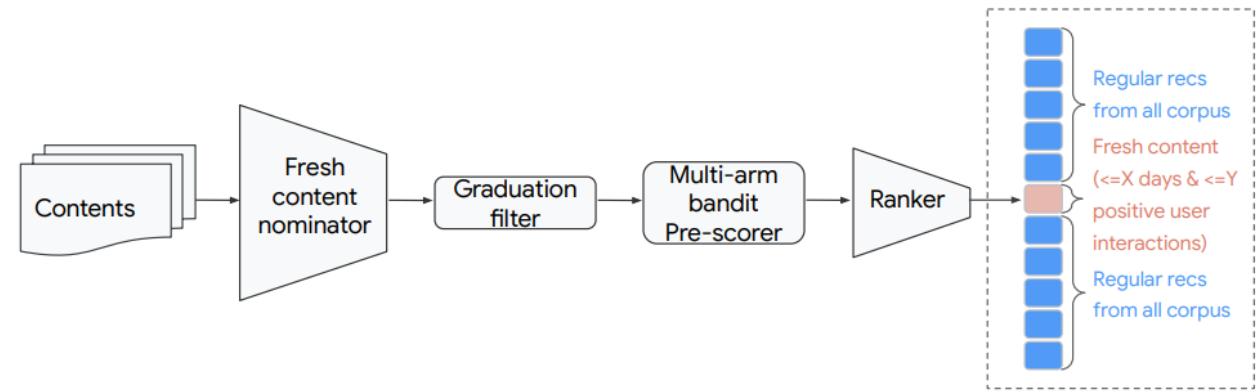
## Устаревшие рекомендации

Пользователи жалуются, что в рекомендациях только устаревший контент. Что делать?

# Устаревшие рекомендации

Пользователи жалуются, что в рекомендациях только устаревший контент. Что делать?

- Шаг первый – добавить источник кандидатов со свежими кандидатами :)



**Figure 2: A dedicated fresh content recommendation stack.**

[Fresh Content Needs More Attention \(YouTube, 2023\)](#)

# Что должна оптимизировать генерация кандидатов?

- Ранжирующая модель выбирает, кто попадёт в выдачу
- Если кандген выдал что-то, что ранкеру не понравится – оно никуда не прорастет
  - Даже если оно хорошее

Нужна согласованность между генерацией кандидатов и ранжированием.

# Выводы

- Начинайте с простых источников кандидатов
- Источников кандидатов всегда несколько
- Нужно уметь их объединять
- Все важные свойства рекомендаций надо обеспечивать начиная с генерации кандидатов
- Recall@K – опасная метрика, согласованность с ранжированием важнее

# Ранжирование

Часть 4

# У нас есть кандидаты. Что дальше?

- Кандидатов больше, чем мы можем показать
- Нужно выбрать и упорядочить
- Это задача стадии ранжирования:
  - Принимает кандидатов и контекст
  - Считает скор для каждого айтема
  - Сортирует по этому скору

# Вход ранжирующей модели

Информация про пользователя, айтем и контекст кодируются в признаки:

- Счётчики (частоты, популярности)
- Контекст (время, устройство)
- Нейросети в основном используют более сырье данные (е.г., историю пользователя)

Feature Type	Example	Feature Engineering
Categorical	User ID / Item ID Brand Main Category	Target Encoding Count Encoding Categorify + Combining Categories
Unstructured list	Keywords Subcategories Colors	Target Encoding Count Encoding Categorify
Numeric	Price Deliver time Avg. reviews	Binning Normalization Gauss Rank
Timestamp	Timestamp	Extract month, weekday, weekend, hour
Timeseries	Events in order Time since last event	# of events in past X Difference in time (lag)
Image	Product image	Extract latent representation with deep learning
Text	Description	Extract latent representation with deep learning
Social graph	Follower/Following graph	Link analysis
Geo location	Addresses	Distances to point of interest

[RecSys 2020 Tutorial: Feature Engineering for Recommender Systems](#)

<https://t.me/WazowskiRecommends/11>

<https://t.me/WazowskiRecommends/16>

# Какая может быть модель

- Линейная модель
- Градиентный бустинг (CatBoost)
- Нейросеть
  - Будет две лекции в курсе на эту тему

# Как учить ранжирующую модель?

Учим ранжирование на impressions:

- Мы знаем, что показали пользователю
- Знаем, как он отреагировал

Позитивы и негативы:

- Позитив – положительная реакция (клик, лайк, заказ, etc)
- Негатив – негативная реакция, либо отсутствие реакции
  - Это impressed negatives

# Pointwise-подход

Каждый показ – отдельный обучающий пример

## Проблемы

- Можно улучшить лосс и не улучшить качество ранжирования
- Модели нужно выучивать масштаб (e.g., средний CTR по пользователю) – это не помогает с ранжированием

# Pairwise-подход

- Берём айтемы, показанные вместе
- Один лучше другого
- Учим позитив быть выше по скору чем негатив
- Фактически оптимизируем ROC-AUC внутри выдачи

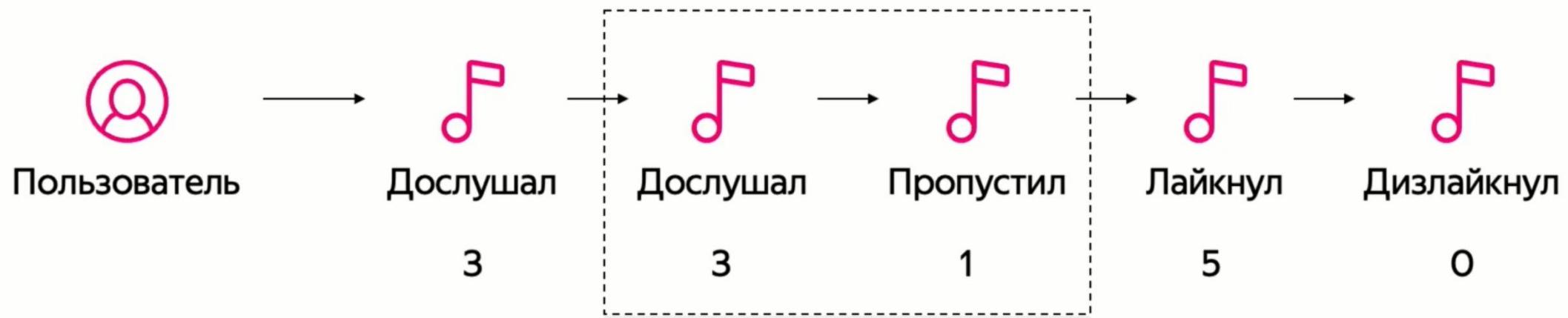
## Лосс

$$- \sum_{p,n \in Pairs} w_{pn} (\log(\sigma(a_p - a_n)))$$

## Взвешенный RankNet

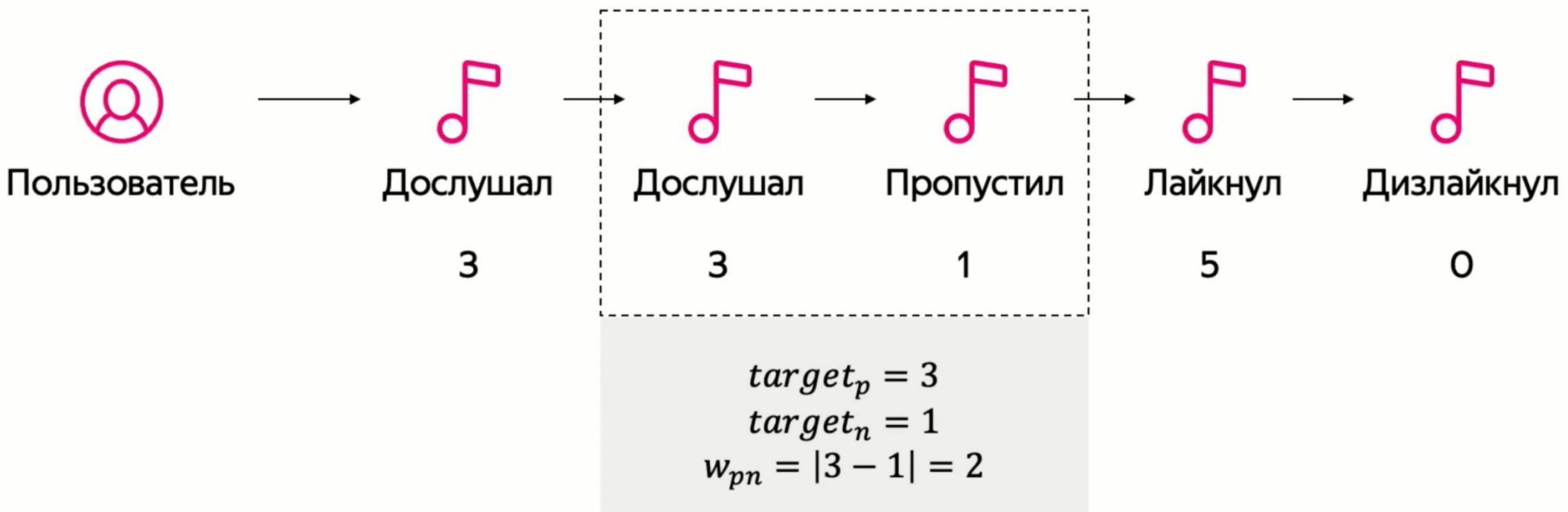
[Как улучшить знакомые подходы для рекомендации незнакомого \(Савва Степурин, Яндекс Музыка\)](#)

# Несколько сигналов



[Как улучшить знакомые подходы для рекомендации незнакомого \(Савва Степурин, Яндекс Музыка\)](#)

# Несколько сигналов



Как улучшить знакомые подходы для рекомендации незнакомого (Савва Степурин, Яндекс Музыка)

# Несколько сигналов

Выкатили в А/В тест новую модель – TLT  
(время прослушивания) выросло, а количество  
лайков уменьшилось. Что делать?

# Несколько сигналов

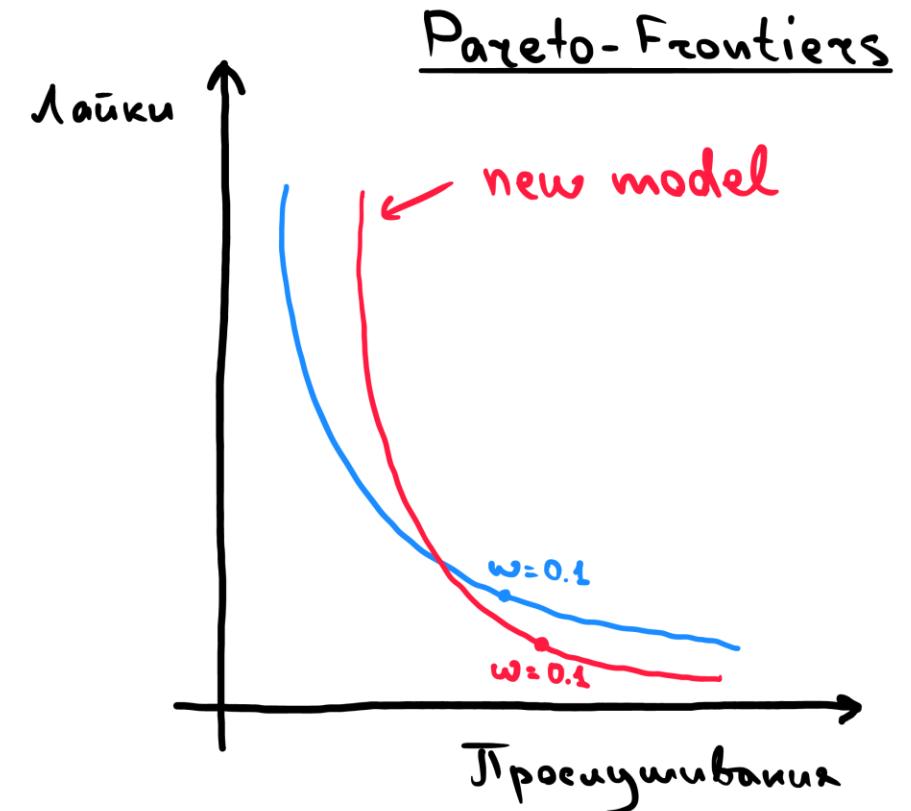
Выкатили в А/В тест новую модель – TLT  
(время прослушивания) выросло, а количество  
лайков уменьшилось. Что делать?

- Переподбирать веса (увеличить вес лайка в обучении)
- Почему так произошло?

# Несколько сигналов

Выкатили в А/В тест новую модель – TLT (время прослушивания) выросло, а количество лайков уменьшилось. Что делать?

- Переподбирать веса (увеличить вес лайка в обучении)
- Почему так произошло? Сдвинулся парето-фронт
- Альтернатива – сделать отдельные модели на каждый тип сигнала



# Twitter, The Algorithm (2023)

The outputs of the model are combined into a final model score by doing a weighted sum across the predicted engagement probabilities. The weight of each engagement probability comes from a configuration file, read by the serving stack [here](#). The exact weights in the file can be adjusted at any time, but the current weighting of probabilities (April 5, 2023) is as follows:

```
scored_tweets_model_weight_fav: 0.5
scored_tweets_model_weight_retweet: 1.0
scored_tweets_model_weight_reply: 13.5
scored_tweets_model_weight_good_profile_click: 12.0
scored_tweets_model_weight_video_playback50: 0.005
scored_tweets_model_weight_reply_engaged_by_author: 75.0
scored_tweets_model_weight_good_click: 11.0
scored_tweets_model_weight_good_click_v2: 10.0
scored_tweets_model_weight_negative_feedback_v2: -74.0
scored_tweets_model_weight_report: -369.0
```

Essentially, the formula is:

```
score = sum_i { (weight of engagement i) * (probability of engagement i) }
```

<https://github.com/twitter/the-algorithm-ml/tree/main/projects/home/recap>

# X, The Algorithm (2026)

## Scoring and Ranking

The Phoenix Grok-based transformer model predicts probabilities for multiple engagement types:

```
Predictions:  
|--- P(favorite)  
|--- P(reply)  
|--- P(repost)  
|--- P(quote)  
|--- P(click)  
|--- P(profile_click)  
|--- P(video_view)  
|--- P(photo_expand)  
|--- P(share)  
|--- P(dwell)  
|--- P(follow_author)  
|--- P(not_interested)  
|--- P(block_author)  
|--- P(mute_author)  
|--- P(report)
```

The **Weighted Scorer** combines these into a final score:

```
Final Score = Σ (weight_i × P(action_i))
```

Positive actions (like, repost, share) have positive weights. Negative actions (block, mute, report) have negative weights, pushing down content the user would likely dislike.

# Оценка качества ранжирования

## Информационный поиск != классификация:

- Для оценки качества классификации применяют те же метрики, но ранжируют все объекты со всеми
- В рекомендациях (и информационном поиске) ранжируем только внутри осмысленных групп (выдач, сессий, пользователей)

... И не забываем про temporal split!

## Метрика качества

$$\frac{\sum_{p,n \in Pairs} w_{pn} [a_p > a_n]}{\sum_{p,n \in Pairs} w_{pn}}$$

Взвешенная доля правильно угаданных пар

[Как улучшить знакомые подходы для рекомендации незнакомого \(Савва Степурин, Яндекс Музыка\)](#)

# Дообучение

- Если не переобучать / дообучать модели на новых данных, качество рекомендаций упадёт
- Обычно модели переобучают минимум раз в неделю
- Челлендж – дообучаться при использовании фидбека с разной степенью задержки

# Яндекс Музыка



# Research & Development

Часть 5

# Чем занималась наша RecSys R&D команда в Яндексе

- Поиск новых технологий
- Экспериментальная деятельность
- Поддержка и развитие R&D инструментов
- Внедрения

# Поиск новых технологий

- Постоянное изучение arXiv, конференций, воркшопов, инженерных и ресерч блогов
  - Чтение и поиск статей обсудим на третьем семинаре
- Потребность в R&D реализуется с двух сторон
  - Мы сами “приносили” новые технологии для улучшения рекомендаций в продуктах
  - Продукты приходили со своими проблемами
- Для нас первый сценарий реализовывался чаще, поэтому поиск новых технологий был очень важен

# Экспериментальная деятельность

- Прототипирование – выдвигаем и проверяем различные гипотезы
  - “Если сделаем X, улучшим базовое качество рекомендаций на Y, потому что Z”
- Метрика успешности – “пропускная способность” в гипотезах
  - количество выдвинутых / проверенных / успешных гипотез
- Доля успешных гипотез растёт вместе с интуицией
- Обсудим позже в курсе

# Поддержка и развитие R&D инструментов

- Фреймворк для обучения нейросетей
- Работа с данными
- Инструменты для внедрений
- Командные процессы

# Фреймворк для обучения нейросетей

- Самый важный компонент, больше всего влияющий на проверку гипотез
- Предела совершенству нет:
  - Как конфигурировать обучения
  - Как должен выглядеть конструктор модели
  - Какие нужны колбеки и что логировать
  - Поддержка распределенного обучения
  - Чтение большого датасета по сети
- Обсудим фреймворки на семинаре про обучение нейросетей

# Работа с данными

- Самые профитные гипотезы почти всегда связаны с данными
  - Подать на вход в модель что-то принципиально новое
  - Изменить представление входных данных
  - Модифицировать целевую задачу
  - Придумать новую процедуру предобучения
- Нам доступны триллионы пользовательских событий
- Нужен удобный, гибкий и быстрый фреймворк для работы с данными
- Важно самим управлять обработкой данных (а не делегировать другим инженерам)

# Инструменты для внедрений

- Быстрые “применялки” моделей
- Регулярные процессы дообучения
- Сервисы для применения моделей
- Оффлайн настройки векторов
- Индексы для генерации кандидатов
- Что-то делали сами, с чем-то нам помогали  
более инфровые команды

# Командные процессы

- Выстраивание процессов в R&D сложнее, чем в обычной разработке
- Очень много неопределенности – никогда точно не знаешь чем закончатся эксперименты
- Сколько людей должно заниматься одним проектом?
- Как лучше проводить планирования?
- Сколько проектов должно быть у одного человека?
- Как логировать эксперименты? Чтобы через полгода можно было вспомнить что делали
- Как организовать чтение статей?

# Внедрения

- Постоянное общение с продуктovыми командами
- Рассказывали про наши новые технологии и планы
- Договаривались про совместные внедрения
  - С активным участием
  - Или в качестве консультантов
- Делали общие для экосистемы Яндекса инструменты
- Кросс- опыление: узнавали от сервисов много нового и рассказывали другим сервисам
- Круто видеть импакт на продукты, которыми пользуется много людей (Музыка, Кинопоиск, Алиса, Поиск, Маркет, etc)

# Итоги лекции

Научились строить рекомендательную систему с нуля:

- Выбираем целевые метрики
- Строим обработку данных (логи)
- Готовим источники кандидатов
- Обучаем ранжирующую модель

И (возможно) обсудили R&D.

# Дополнительные материалы

- Тг-канал [@WazowskiRecommends](#)
  - канал Миши Ройзнера, создателя основного рекомендательного движка Яндекса
- [Тренировки по ML 2.0 Лекция 7: Рекомендательные системы](#)
  - Дизайн рекомендаций Яндекс Музыки от Саввы Степурина, разработчика Яндекс Музыки
- [Как улучшить знакомые подходы для рекомендации незнакомого](#)
  - Наглядный доклад от Саввы про улучшение рекомендательного стека Музыки

# На семинаре

- Популярные рекомендательные бейзлайны
- Разбор статьи про Yambda