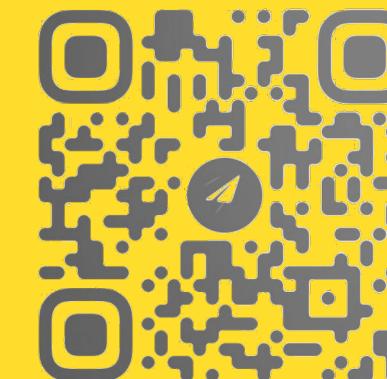


# Рекомендательные системы

2023/2024



@ANANYEVAME

**Марина Ананьева**

Head of RecSys @ Tinkoff,  
PhD student

email: [mananeva@hse.ru](mailto:mananeva@hse.ru)

# Course outline

---

Неделя	Тема	Подробнее:
1	<b>Введение в рекомендательные системы</b>	Формализация задачи ранжирования. Типы (implicit, explicit), форматы (е.г.матрица интеракций) данных. Функции и метрики качества ранжирования. Классификация рекомендательных моделей.
2	<b>Матричные факторизации и колаборативная фильтрация</b>	Матричные факторизации (SVD++ и др.). Проблемы большой размерности и разреженности данных и холодного старта. Коллаборативная фильтрация (на примере ALS).
3	<b>Контентные и гибридные модели</b>	Контентные рекомендательные подходы (на примере LightFM, DSSM). Виды triplet loss и negative sampling для RecSys. Гибридные подходы (на примере каскадной архитектуры) и особенности оценки качества.

# Course outline

Неделя	Тема	Подробнее:
4	<b>Нейросетевые модели на последовательностях данных</b>	Sequential, session, context-based подходы рекомендательных систем. Задачи next-basket и next-item предсказаний. Учет временного контекста в моделях.
5	<b>Автоэнкодеры и вариационные автоэнкодеры</b>	Адаптация автоэнкодеров и вариационных автоэнкодеров в задачах рекомендательных систем (VAE, Mult-VAE, Mult-DAE, RecVAE). Достоинства и недостатки всех рассмотренных ранее подходов.
6	<b>Graph и knowledge-graph подходы. Объяснения рекомендаций</b>	Трансдуктивные и индуктивные модели. Проблема out-of-sample пользователей и объектов. Графовые рекомендательные системы (GNN, GCN, GraphSage, PinSage, GAT). Model-intrinsic и model-agnostic подходы интерпретации рекомендательных систем. Shapely values. Интерпретация весов внутри модели (пример iALS), через механизм внимания. Обзор нейросетевых подходов для рекомендаций с объяснениями.
7	<b>Написание сервиса рекомендательной системы</b>	Разбор и имплементация простой архитектуры сервиса рекомендательной системы. Использование Docker, FAST API, Inference server (Triton), Redis для поднятия сервиса. Обзор подходов и архитектур для рекомендательных систем на больших данных и высоконагруженных системах.

# Оценивание

**Final grade = 0.3 \* Home Assignment + 0.15 \* Article Summary + 0.25 \* Weekly Quizzes + 0.3 \* Exam (Case-study)**

где:

- **Home Assignments** - 1 домашнее задание в Jupyter Notebook (max 10 баллов).
- **Article Summary** - критический анализ статьи из предложенного списка с ACM RecSys 2023 (max 10 баллов).
- **Weekly Quizzes** - 6 квизов по мотивам материалов семинаров, которые сдаются перед началом следующего занятия в Google Forms (ариф.среднее за все квизы, max 10 баллов за каждый).
- **Exam** - письменный экзамен в формате решения case-study построения рекомендательной системы для бизнеса (max 10 баллов).

# Полезные ссылки

Чат курса в Telegram

<https://t.me/+9MN5JL4xj5xjYTBi>



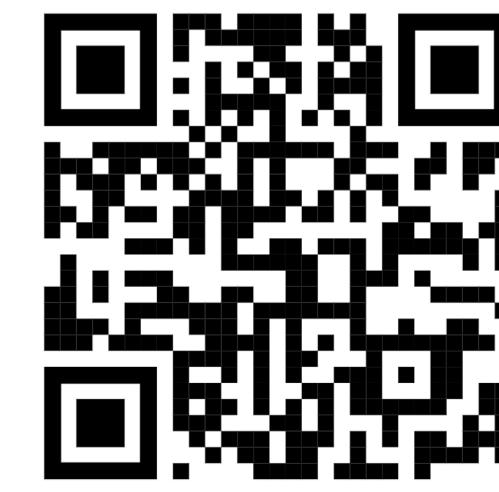
Таблица с оценками

[ссылка на docs.google](#)



Wiki с Пудом и материалами

[http://wiki.cs.hse.ru/RecSys\\_2023](http://wiki.cs.hse.ru/RecSys_2023)



Github репа

[https://github.com/anamarina/RecSys\\_course](https://github.com/anamarina/RecSys_course)



# Introduction

# **Рекомендательные системы и персонализация**

**Персонализация** - это набор различных способов учета интересов и потребностей пользователей на основе имеющихся о них данных.

**Рекомендательные системы** - это тип систем информационной фильтрации (IR), цель которых - выдавать каждому пользователю наиболее релевантные ему объекты. Может рассматриваться как:

- \* ML и не-ML задача
- \* С персонализацией и без персонализации

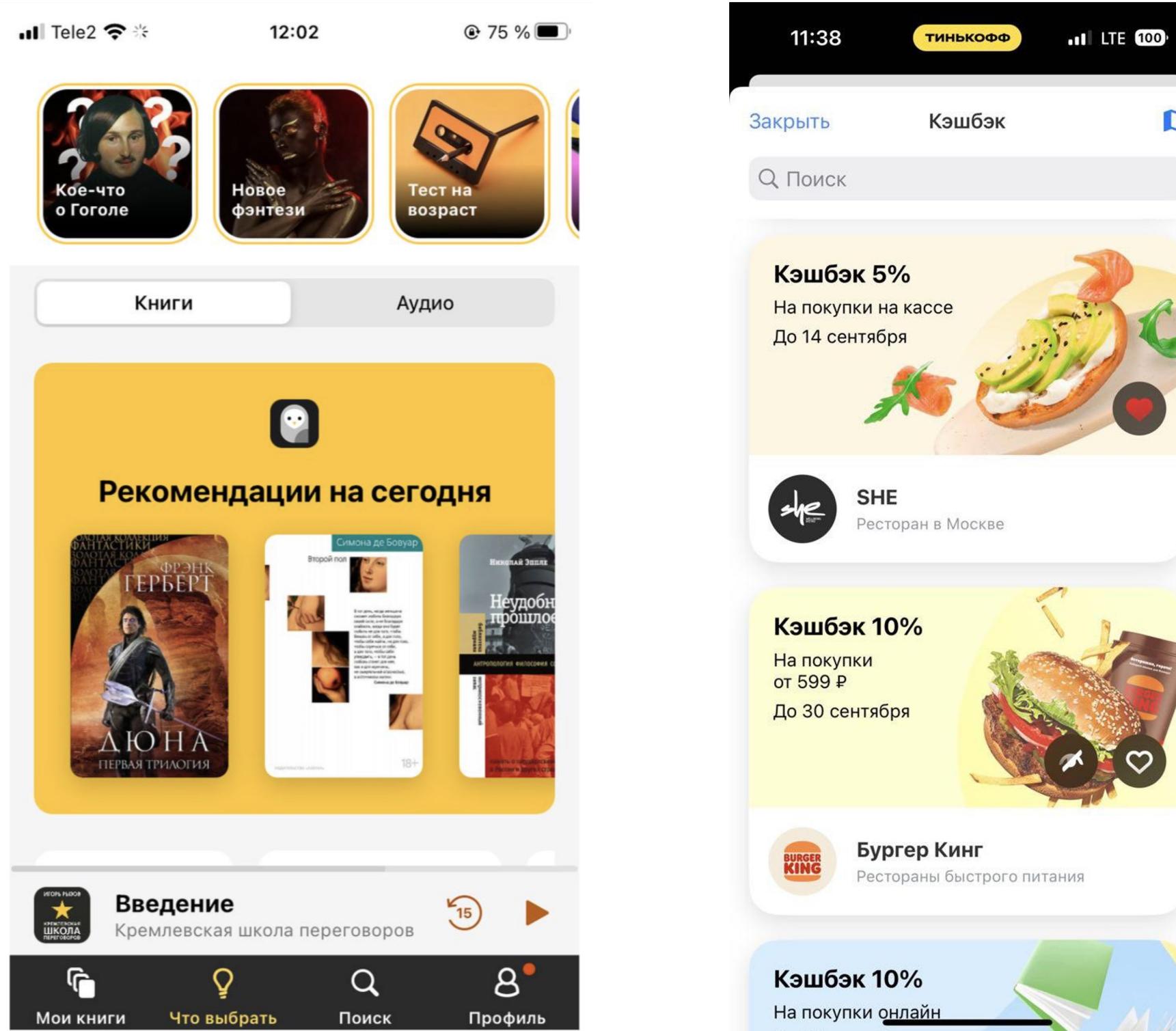
# Зачем рекомендации **пользователю**?

-  Более релевантные контент, акции, продукты, коммуникации в сервисе
-  Снижение негатива (от лишних коммуникаций, нерелевантных предложений)
-  Положительный customer experience (удобство использования сервиса, интерес)
-  Экономия времени на поиск внутри сервиса
-  Готовность и интерес к exploration, когда есть актуальный контекст и понятные ожидания

# Зачем рекомендации пользователю?

- Более релевантный контент, акции, продукты, коммуникации и т.д.

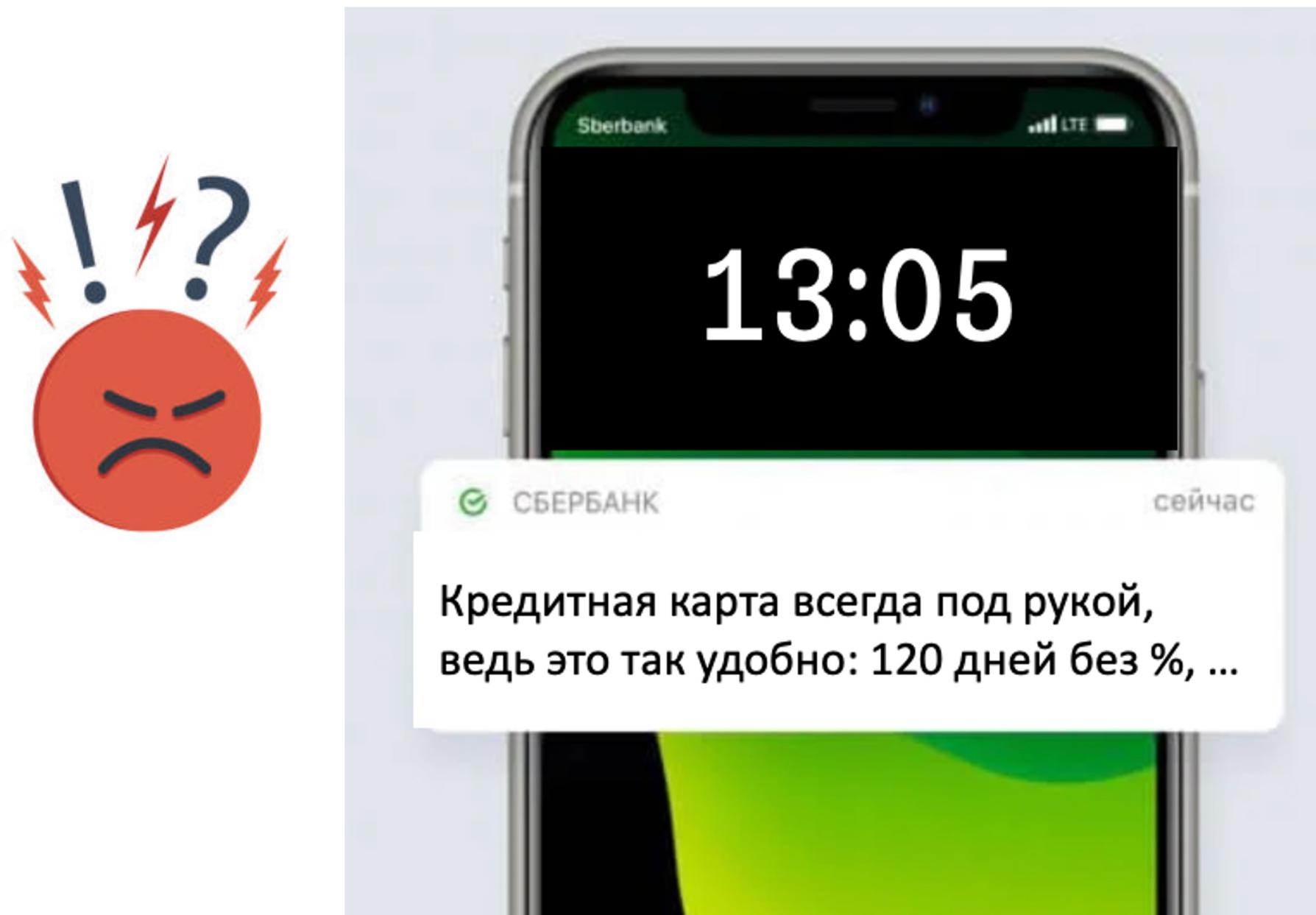
Примеры: рекомендации товаров в корзине Ozon, рекомендации книг в MyBook, персональные акции в Dodo



# Зачем рекомендации пользователю?

- Снижение негатива (удержания лишних коммуникаций, нерелевантных предложений)

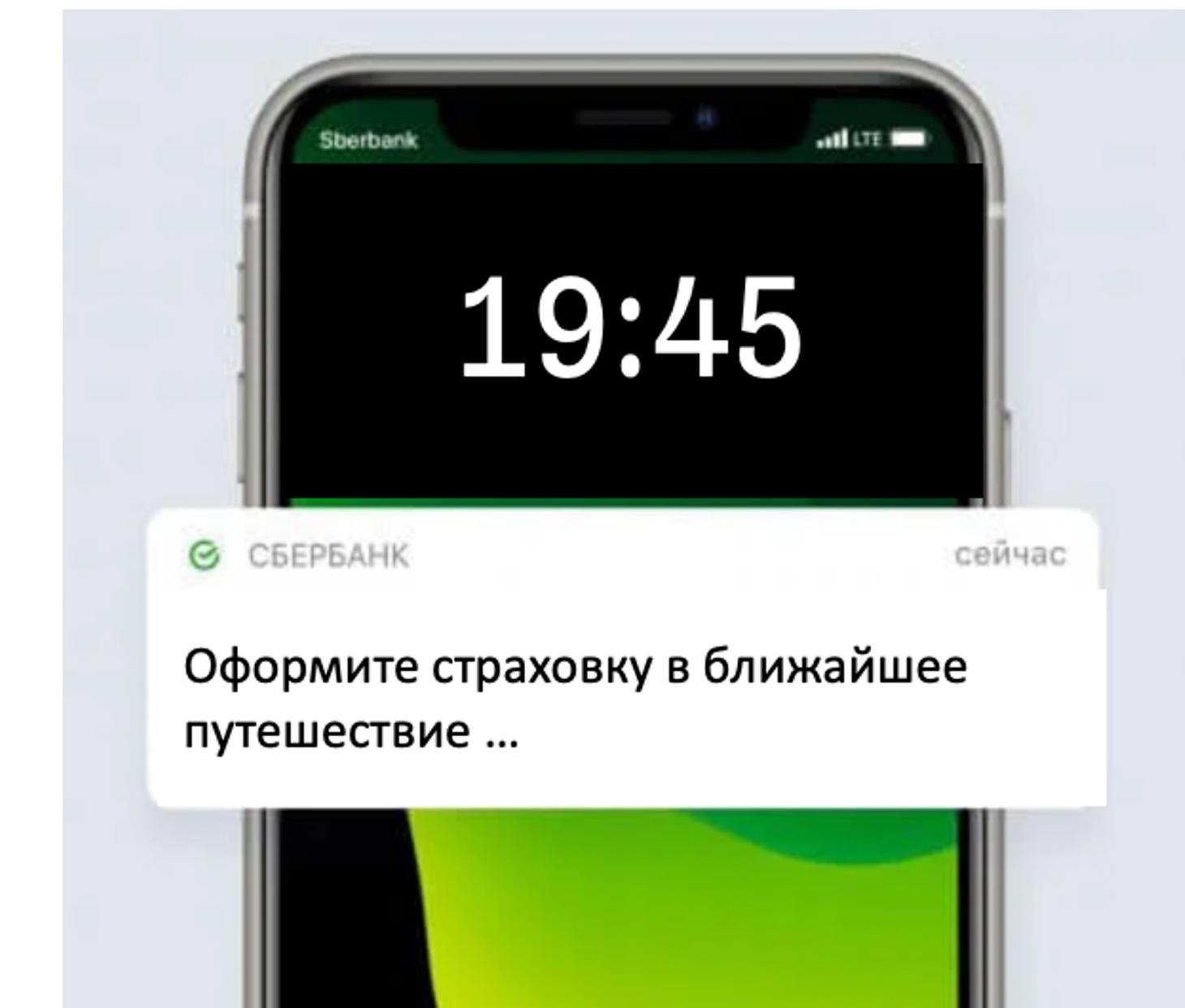
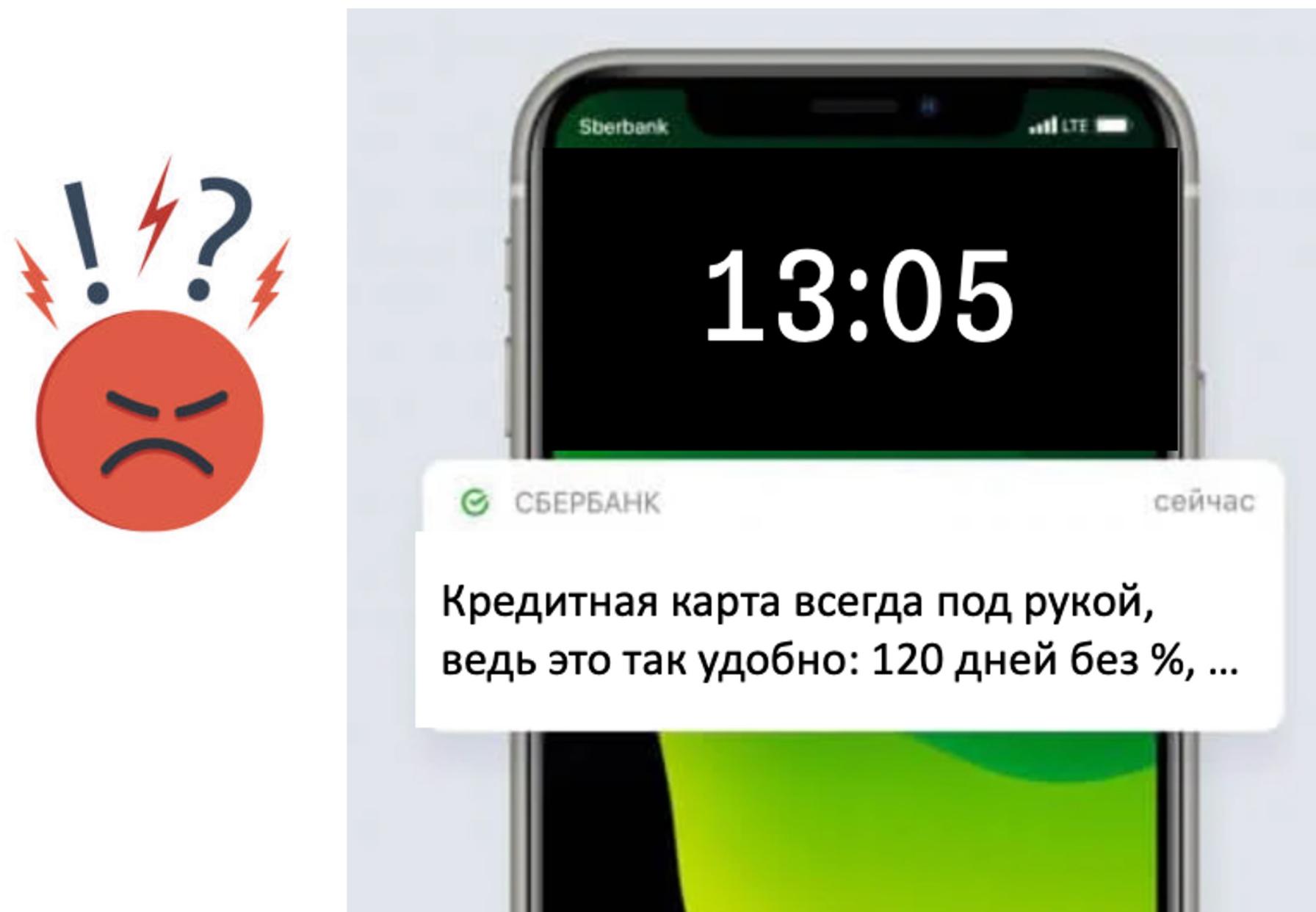
**Примеры:** отбор таргетной аудитории для маркетинговой акции, прогнозирование персонального времени отправки пуша, ранжирование лучших каналов для коммуникаций, ранжирование маркетинговых акций.



# Зачем рекомендации пользователю?

- Снижение негатива (удержания лишних коммуникаций, нерелевантных предложений)

**Примеры:** отбор таргетной аудитории для маркетинговой акции, прогнозирование персонального времени отправки пуша, ранжирование лучших каналов для коммуникаций, ранжирование маркетинговых акций.



# Зачем рекомендации пользователю?



- Положительный customer experience (повышается удобство использование сервиса)

**AVITO.ru**  
легко купить, легко продать

**Самый большой сайт бесплатных объявлений**

Частные объявления и объявления компаний о покупке и продаже. Домашние животные, бытовая техника, электроника, недвижимость, автомобили, одежда и многое другое. На сайте **1 187 102 объявления**

**ПОДАТЬ ОБЪЯВЛЕНИЕ**

[Зарегистрироваться](#) | [Войти](#)

**Восточная часть »**

**Москва** **Санкт-Петербург** **Екатеринбург** **Казань** **Краснодар**

**Нижний Новгород** **Ростов-на-Дону** **Самара** **Уфа** **Челябинск**

Астраханская обл. Новгородская обл.  
Башкортостан Оренбургская обл.  
Белгородская обл. Пензенская обл.  
Брянская обл. Пермский край  
Владимирская обл. Псковская обл.  
Волгоградская обл. Ростовская обл.  
Вологодская обл. Рязанская обл.  
Воронежская обл. Самарская обл.  
Ивановская обл. Саратовская обл.  
Калининградская обл. Свердловская обл.  
Калужская обл. Смоленская обл.  
Кировская обл. Ставропольский край  
Краснодарский край Тамбовская обл.  
Курганская обл. Татарстан  
Курская обл. Тверская обл.  
Ленинградская обл. Тульская обл.  
Липецкая обл. Удмуртия  
Марий Эл Ульяновская обл.  
Московская обл. Челябинская обл.  
Мурманская обл. Чувашия  
Нижегородская обл. Ярославская обл.

[Сделать стартовой](#)

**Avito** Авто Недвижимость Работа Услуги Ещё

Любая категория Поиск по объявлениям Москва

только в названиях  только с фото  сначала из Москвы

[Авто и «ЛизАлерт»](#) [Куда сходить в Ярославле](#) [Топ ЖК Москвы](#) [Где пожить в Санкт-Петербурге](#) [Российские бренды косметики](#) [Налоговый вычет по ипотеке](#) [Приводим участок в порядок](#) [Три истории о работе](#) [Как въехать](#)

**Рекомендации для вас**

 Щенки сибирской хаски 15 000 ₽ Москва, Молодёжная 22 августа 10:12	 Шкаф 9 990 ₽ Москва, Академическая Сегодня 11:02	 Девочка и мальчик вельш корги РКФ 75 000 ₽ Москва, Бунинская аллея 19 августа 15:20	 Стеллаж Новый аналог Икеа Каллакс 8 ячеек 3 600 ₽ Москва, Бутырская Вчера 12:03
 20 Огромных Android TV 9.0. Wi-Fi 50 31 000 ₽ Москва, Молодёжная 19 августа 12:01	 Щенки Вельш-корги Пемброк 45 000 ₽ Москва, Беломорская 17 августа 14:15	 Микро черные мыши шпицузли комочки счастья 95 000 ₽ Москва, Охотный ряд 21 августа 00:25	 Деревянный стеллаж 1 200 ₽ Москва, Печатники 18 августа 11:45

# Зачем рекомендации пользователю?

- Экономия времени на поиск внутри приложения/платформы

Google Scholar

The screenshot shows the Google Scholar search interface. At the top, there is a search bar with a blue 'Search' button. Below it are two radio buttons: 'Articles' (selected) and 'Case law'. The main area is titled 'Recommended articles' and contains three article cards:

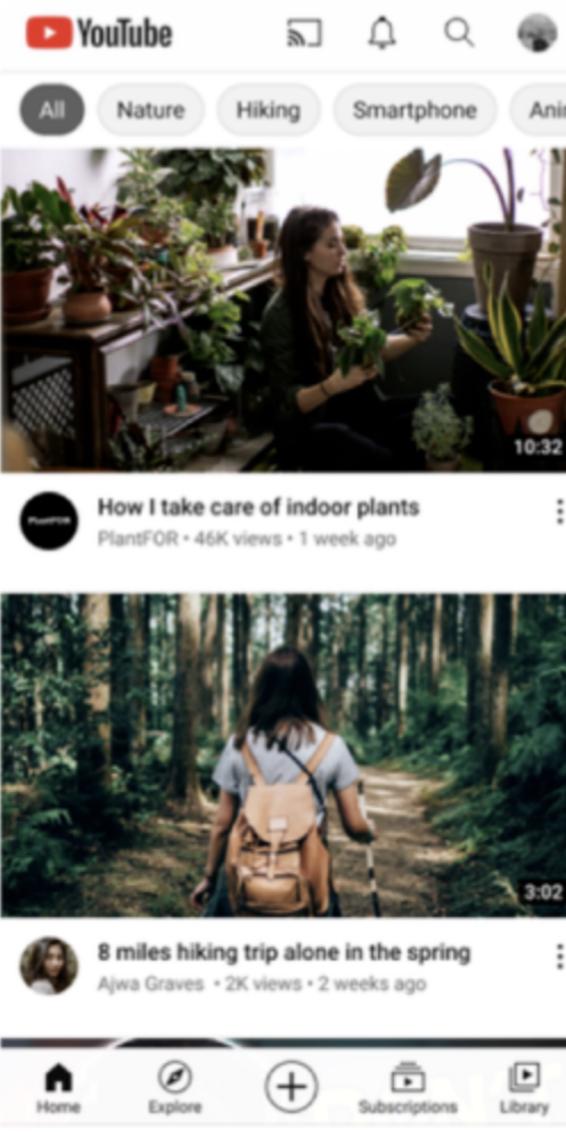
- Knowledge graph preference migration network for recommendation**  
R Ma, X Bu, Z Chen, H Wu, Y Ma, L Zhao  
Expert Systems with Applications - 3 days ago
- KSGAN: Knowledge-aware subgraph attention network for scholarly community recommendation**  
Q Lu, W Du, W Xu, J Ma  
Information Systems - 4 days ago
- Multi-temporal Sequential Recommendation Model Based on the Fused Learning Preferences**  
J Chen, L Pan, S Dong, T Yu, L Xiao, M Yao, S Luo  
International Journal of Computational Intelligence Sys... - 6 days ago [HTML](#)

Below the first article card, there is a link 'More articles from 3 days ago'.

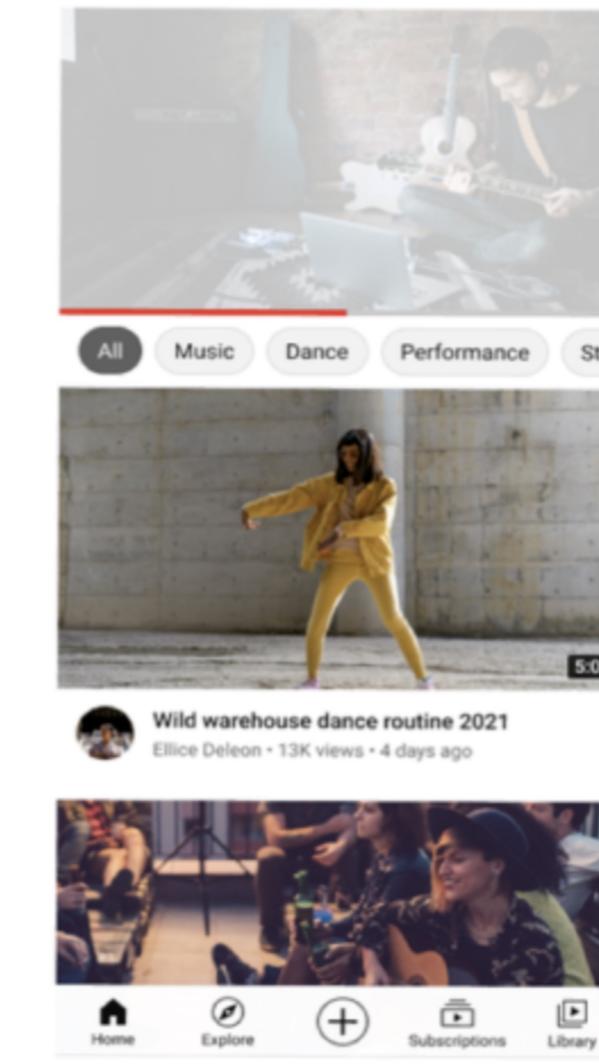
# Зачем рекомендации пользователю?

- **Discovery.** Удовлетворение потребностей клиентов, готовых к exploration (чему-то новому и интересному через рекомендации сервиса)

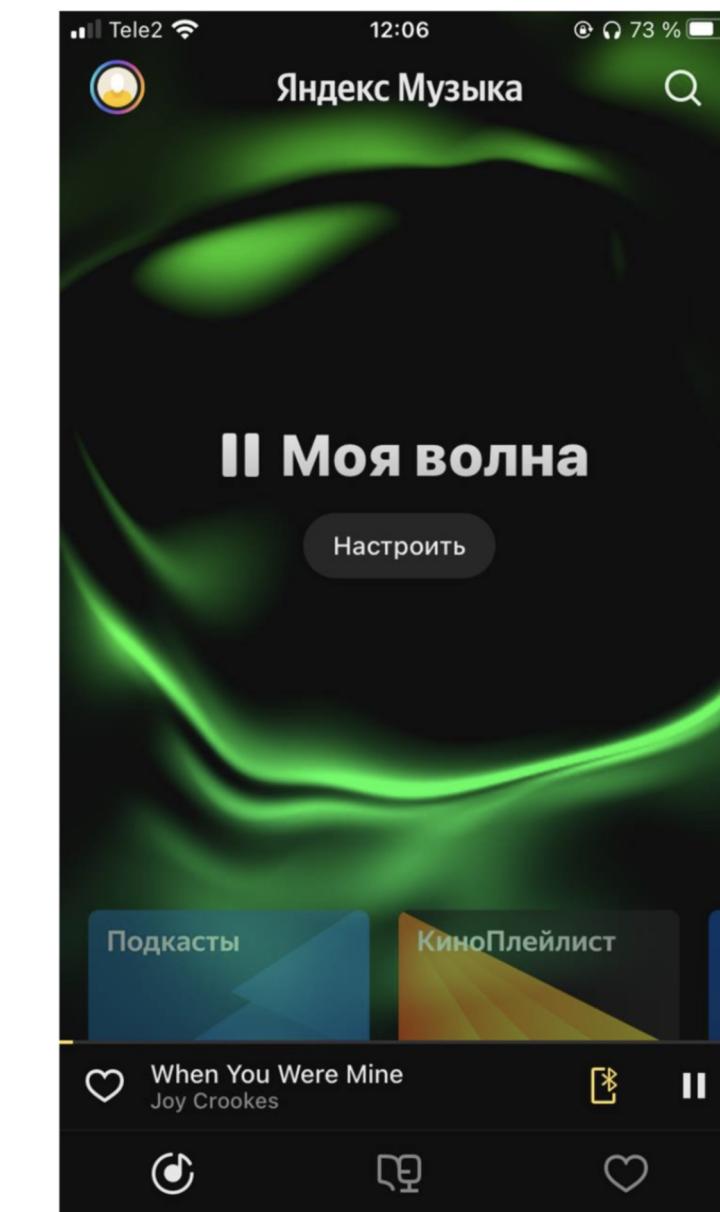
**Примеры:** рекомендации видео на YouTube, рекомендации контента в TikTok/Instagram/etc., рекомендации музыки в Spotify и Яндекс.Музыке.



Recommendations on homepage.



Recommendations on "Up next."



# Зачем рекомендации **бизнесу?**

# Зачем рекомендации **бизнесу**?

- ↑ Увеличение целевых бизнес-метрик (конверсии в заказы, клики, сохранения в избранное, подписки)
- ↑ Экономия на лишних коммуникациях, костах на привлечение, x-sell, акции
- ↑ Наращивание лояльности, NPS
- ↑ Дополнительная монетизация (uplift к бизнес-метрикам, подписка за использование сервиса, плата за доп.фичи)

# Формализация задачи

# Постановка задачи ранжирования

$X$  - множество объектов  
для рекомендаций

$X^l = \{x_1, \dots, x_l\}$  - обучающая  
выборка из  
объектов

$i < j$  на парах  $(i, j) \in \{1, \dots, l\}^2$

Необходимо построить  $a: X \rightarrow \mathbb{R}$  такую,

что:  $i < j \Rightarrow a(x_i) < a(x_j)$

Задача частичної правильного порядок

# Пример 1. Ранжирование поисковой выдачи

$\mathcal{D}$  - множество документов

$Q$  - множество запросов

$\mathcal{R}_q \subseteq \mathcal{D}$  - мн.во документов к запросу  $q$

$X = Q \times \mathcal{D}$  - объект - это пары (запрос, документ)

$x \equiv (q, d), q \in Q, d \in \mathcal{R}_q$

$y$  - упорядоченное мн.во рейтингов

(например, ассесорская разметка или click rate)

$y: X \rightarrow Y$  оценки relevance

Правильный порядок док-в к запросу  $q$ :

$(q, d_1) < (q, d_2) \Leftrightarrow y(q, d_1) < y(q, d_2)$

## Пример 2. Рекомендации пользователям

$U$  - множество пользователей,  $u \in U$

$I$  - множество объектов (аттракторов),  $i \in I$

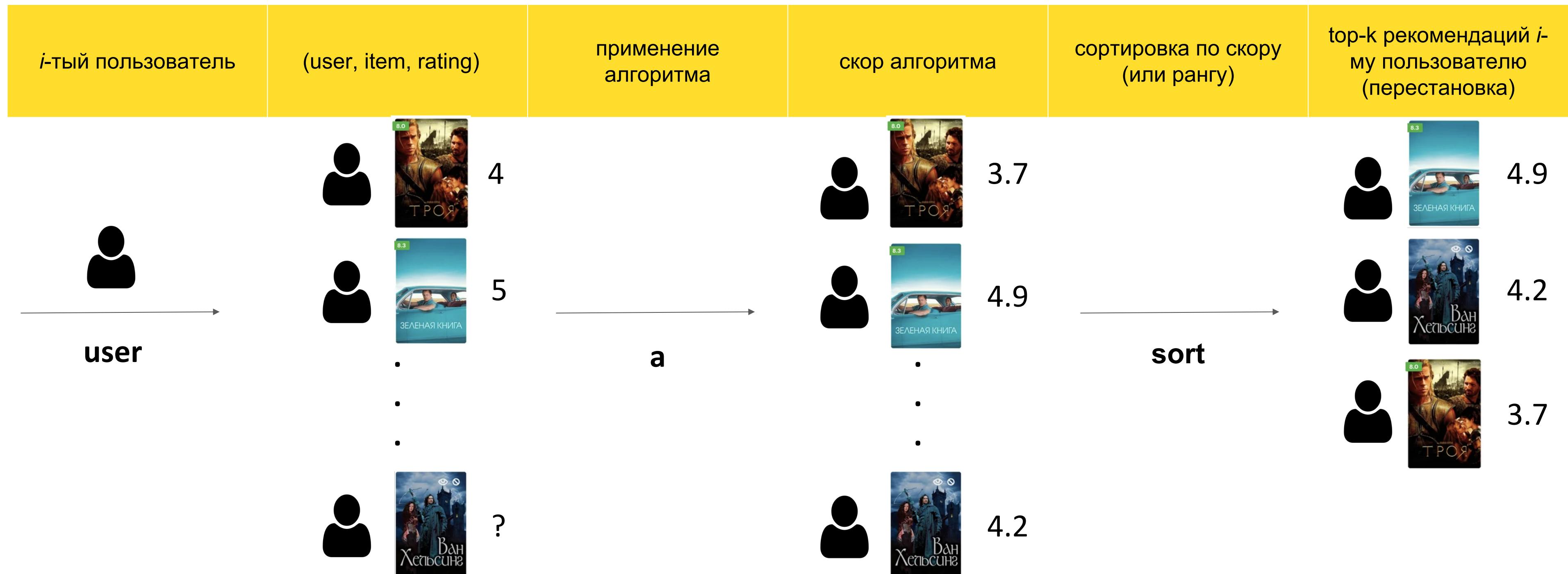
$X = U \times I$  - пары (пользователь, аттрактор)  
 $(u, i)$

$$(u, i_1) < (u, i_2) \Leftrightarrow y(u, i_1) < y(u, i_2)$$

Список аттракторов  $i$ , упорядоченных функцией  
 $y(u, i)$ , называется рекомендациями  
пользователю  $u$ .

# Пример 2. Рекомендации пользователям

$$R = \{(u, i, r_{iu})\}$$



# Типы данных

# Explicit & implicit данные

**Explicit feedback / Явные данные** - тип данных, когда известны оценки пользователей в результате взаимодействия с объектом. Например, датасет с оценками пользователей по 5-балльной шкале, насколько им понравились просмотренные фильмы.

**Implicit feedback / Неявные данные** - тип данных, когда нам неизвестно насколько понравился или не понравился объект после того, как пользователь провзаимодействовал с ним.

**Матрица интеракций** - это матрица, у которой по одной оси - id пользователей, по второй оси - id объектов, а на пересечении - оценка/наличие взаимодействия.

	item 1	....	item N
user 1	0	1	...
....	1	0	...
user M	...	...	1

implicit ?

	item 1	....	item N
user 1	0	3	...
....	5	0	...
user M	...	...	1

explicit ?

# Explicit & implicit данные

**Explicit feedback / Явные данные** - тип данных, когда известны оценки пользователей в результате взаимодействия с объектом. Например, датасет с оценками пользователей по 5-балльной шкале, насколько им понравились просмотренные фильмы.

**Implicit feedback / Неявные данные** - тип данных, когда нам неизвестно насколько понравился или не понравился объект после того, как пользователь провзаимодействовал с ним.

```
'''
```

Пример 1. По 5 клиентам магазина известна история покупок из 10 уникальных товаров.

1 – товар был куплен, 0 – не был.

```
'''
```

```
import numpy as np
from numpy import random

random.randint(low=0, high=2, size=(5,10))

array([[0, 0, 0, 0, 0, 1, 1, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 1, 1, 1, 0, 1],
       [1, 0, 0, 0, 0, 1, 1, 1, 0],
       [0, 1, 1, 0, 0, 1, 0, 0, 0]])
```

# Explicit & implicit данные

**Explicit feedback / Явные данные** - тип данных, когда известны оценки пользователей в результате взаимодействия с объектом. Например, датасет с оценками пользователей по 5-балльной шкале, насколько им понравились просмотренные фильмы.

**Implicit feedback / Неявные данные** - тип данных, когда нам неизвестно насколько понравился или не понравился объект после того, как пользователь провзаимодействовал с ним.

```
'''  
Пример 2. По 7 пользователям и 5 постам в соц.сети известна история реакций.  
0 - посту поставлен дислайк, 1 - лайк, np.nan - пользователь неставил реакцию у поста.  
'''  
  
num_nans = 10  
matrix = np.random.randint(0, 2, (7, 5)).astype(float)  
matrix.ravel()[np.random.choice(matrix.size, num_nans)] = np.nan  
matrix  
  
array([[ 0.,  1., nan,  0.,  0.],  
       [ 0.,  0.,  1., nan,  1.],  
       [ 1.,  0.,  0.,  1.,  0.],  
       [ 1.,  1.,  1.,  0.,  1.],  
       [ 0., nan,  1.,  1.,  0.],  
       [ 1., nan,  0., nan,  0.],  
       [nan, nan, nan, nan,  0.]])
```

# Explicit & implicit данные

**Explicit feedback / Явные данные** - тип данных, когда известны оценки пользователей в результате взаимодействия с объектом. Например, датасет с оценками пользователей по 5-балльной шкале, насколько им понравились просмотренные фильмы.

**Implicit feedback / Неявные данные** - тип данных, когда нам неизвестно насколько понравился или не понравился объект после того, как пользователь провзаимодействовал с ним.

```
'''
```

Пример 3. По 3 пользователям и 8 ресторанам известна история отзывов на рестораны и их рейтинг по 5-балльной шкале.  
1 – самая низкая оценка, ресторан очень не понравился, 5 – отличная оценка, np.nan – нет рейтинга и отзыва.

```
'''
```

```
matrix = np.random.randint(1, 6, (3, 8)).astype(float)
matrix.ravel()[np.random.choice(matrix.size, num_nans)] = np.nan
matrix

array([[ 1.,  3.,  3.,  4., nan,  5., nan, nan],
       [nan, nan, nan,  3.,  4.,  4.,  4.,  2.],
       [nan, nan,  2.,  1.,  4., nan,  5.,  4.]])
```

# Explicit & implicit данные

**Explicit feedback / Явные данные** - тип данных, когда известны оценки пользователей в результате взаимодействия с объектом. Например, датасет с оценками пользователей по 5-балльной шкале, насколько им понравились просмотренные фильмы.

**Implicit feedback / Неявные данные** - тип данных, когда нам неизвестно насколько понравился или не понравился объект после того, как пользователь провзаимодействовал с ним.

...

**Пример 4.** По 5 клиентам есть градация по разным типам действий.

0.1 – просмотр

0.3 – просмотр + клик

0.5 – просмотр + клик + лайк

0.7 – просмотр + клик + комментарий

1 – просмотр + клик + лайк + комментарий

...

# В каком виде лучше хранить такую матрицу, когда у нас очень много нулей?

```
from scipy.sparse import coo_matrix, csr_matrix

import sys

matrix = random.randint(low=0, high=2, size=(10,10))

print(matrix)
print(f'Sparse матрица: {sys.getsizeof(csr_matrix(matrix))} байтов',
      f'Обычный np.ndarray: {sys.getsizeof(matrix)} байтов',
      sep='\n')
```

```
[[1 1 1 1 0 1 0 1 0 0]
 [0 0 0 0 1 0 1 1 1 0]
 [0 0 1 0 0 1 0 1 1 1]
 [1 1 0 1 1 1 1 0 0 0]
 [0 0 0 0 1 1 0 0 0 1]
 [0 0 1 1 0 1 1 1 0 1]
 [1 0 0 1 0 1 1 1 0 0]
 [1 0 0 0 1 0 1 0 0 1]
 [0 1 1 1 0 1 1 0 1 1]
 [1 0 0 0 0 0 0 0 0 0]]
```

Sparse матрица: 64 байтов

Обычный np.ndarray: 920 байтов

# **Функции ранжирования**

# Функции ранжирования

---

## 1. Pointwise (точечные)

Например, решение через задачу регрессии или классификации.

$$\sum_{i=1}^l (a(x_i) - y(x_i))^2 \rightarrow \min$$

# Функции ранжирования

---

1. Pointwise (точечные)

2. Pairwise (попарные)

$$\sum_{x_i < x_j} [a(x_j) - a(x_i) < 0] \rightarrow \min$$

# Функции ранжирования

---

1. Pointwise (точечные)

2. Pairwise (попарные)

$$\sum_{x_i < x_j} [a(x_j) - a(x_i) < 0] \rightarrow \min$$

$$\sum_{x_i < x_j} L(a(x_j) - a(x_i)) \rightarrow \min$$

# Функции ранжирования

---

1. Pointwise (точечные)

2. Pairwise (попарные)

3. Listwise (списочные)

**RankNet:**

$$\omega := \omega + \eta \frac{1}{1+exp(\langle \omega, x_j - x_i \rangle)} (x_j - x_i)$$

**LambdaRank:**

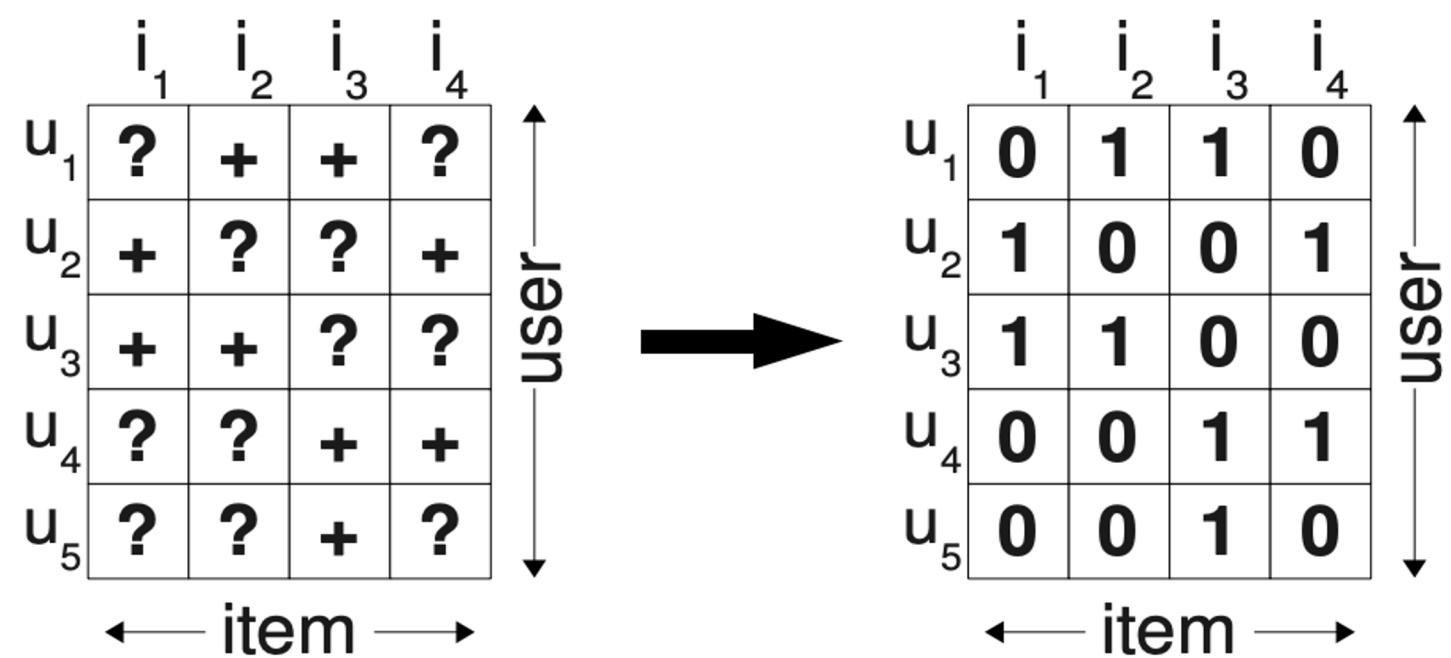
$$\omega := \omega + \eta \frac{1}{1+exp(\langle \omega, x_j - x_i \rangle)} (x_j - x_i) |\Delta nDCG_{ij}| (x_j - x_i)$$

# Функции ранжирования

---

- **BPR** (Bayesian Personalised Ranking, pairwise)
- **WARP** (Weighted Approximate-Rank Pairwise)
- **RankNET** (pairwise)
- **LambdaRANK** (pairwise/listwise)
- **LambdaMART** (pairwise/listwise)

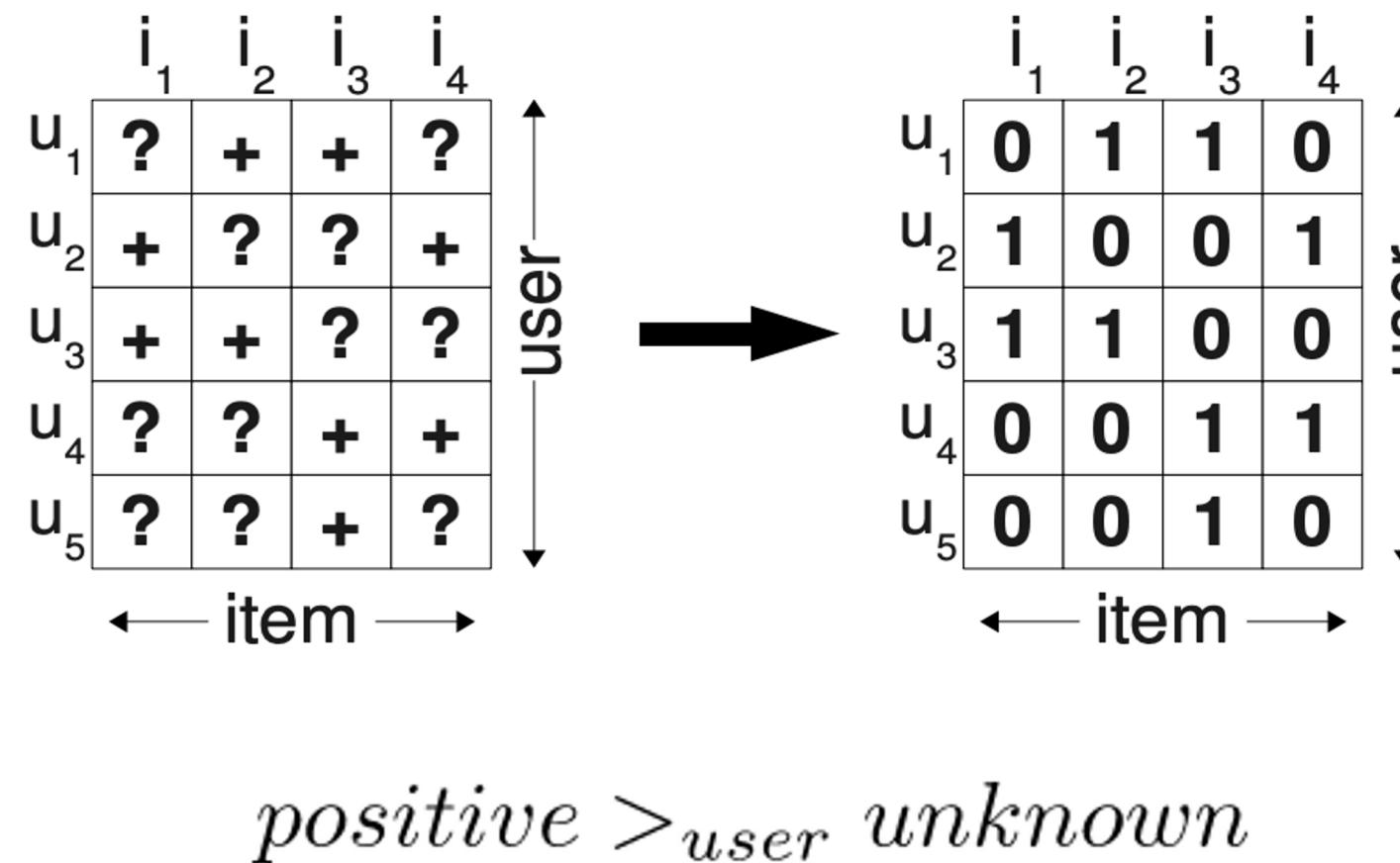
# BPR: Bayesian Personalized Ranking



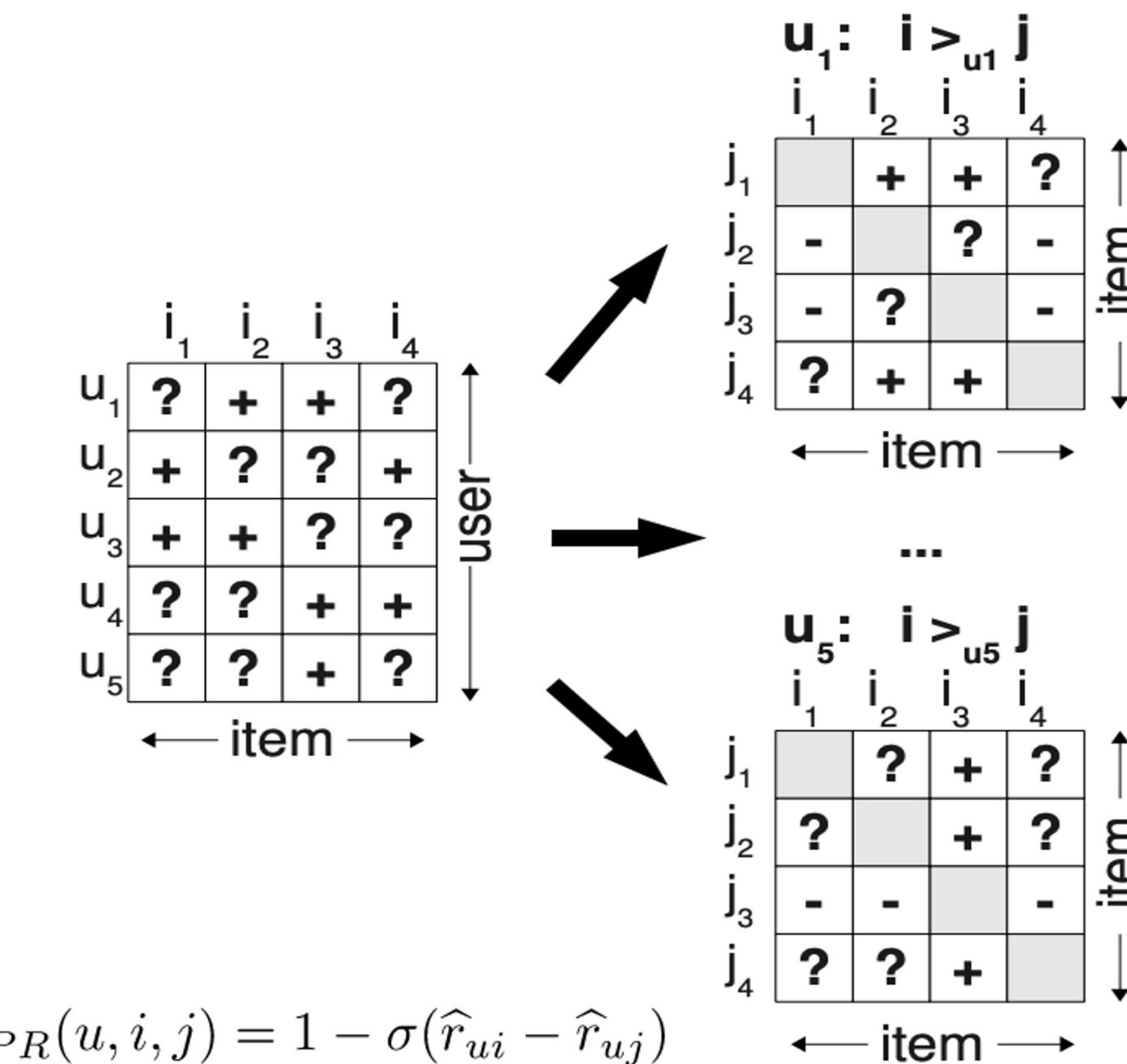
*positive ><sub>user</sub> unknown*

**Рис. 1:** Бинаризация implicit данных, где отсутствие интеракции кодируется как отрицательный пример.

# BPR: Bayesian Personalized Ranking



**Рис. 1:** Бинаризация implicit данных, где отсутствие интеракции кодируется как отрицательный пример.



**Рис. 2:** Попарные предпочтения пользователю по объектам для BPR. + означает, что пользователь предпочитает i объект j-му, знак минуса - наоборот.

# WARP:Weighted Approximate-Rank Pairwise\*

- Тоже смотрим на тройки: **(user, positive\_item, negative\_item)**
- Делаем шаг оптимизации, только если попалась дефектная пара - алгоритм дал выше скор отрицательному примеру, чем положительные
- Добавляем adaptive learning rate: штрафуем больше, если алгоритм ошибается почти сразу.

---

**Algorithm 1** K-os algorithm for picking a positive item.

We are given a probability distribution  $P$  of drawing the  $i^{th}$  position in a list of size  $K$ . This defines the choice of loss function.

Pick a user  $u$  at random from the training set.

Pick  $i = 1, \dots, K$  positive items  $d_i \in \mathcal{D}_u$ .

Compute  $f_{d_i}(u)$  for each  $i$ .

Sort the scores by descending order, let  $o(j)$  be the index into  $d$  that is in position  $j$  in the list.

Pick a position  $k \in 1, \dots, K$  using the distribution  $P$ .

Perform a learning step using the positive item  $d_{o(k)}$ .

---

---

**Algorithm 2** K-os WARP loss

Initialize model parameters (mean 0, std. deviation  $\frac{1}{\sqrt{m}}$ ).

**repeat**

Pick a positive item  $d$  using Algorithm 1.

Set  $N = 0$ .

**repeat**

Pick a random item  $\bar{d} \in \mathcal{D} \setminus \mathcal{D}_u$ .

$N = N + 1$ .

**until**  $f_{\bar{d}}(u) > f_d(u) - 1$  or  $N \geq |\mathcal{D} \setminus \mathcal{D}_u|$

**if**  $f_{\bar{d}}(y) > f_d(u) - 1$  **then**

Make a gradient step to minimize:

$$\Phi\left(\frac{|\mathcal{D} \setminus \mathcal{D}_u|}{N}\right) \max(0, 1 + f_{\bar{d}}(u) - f_d(u)).$$

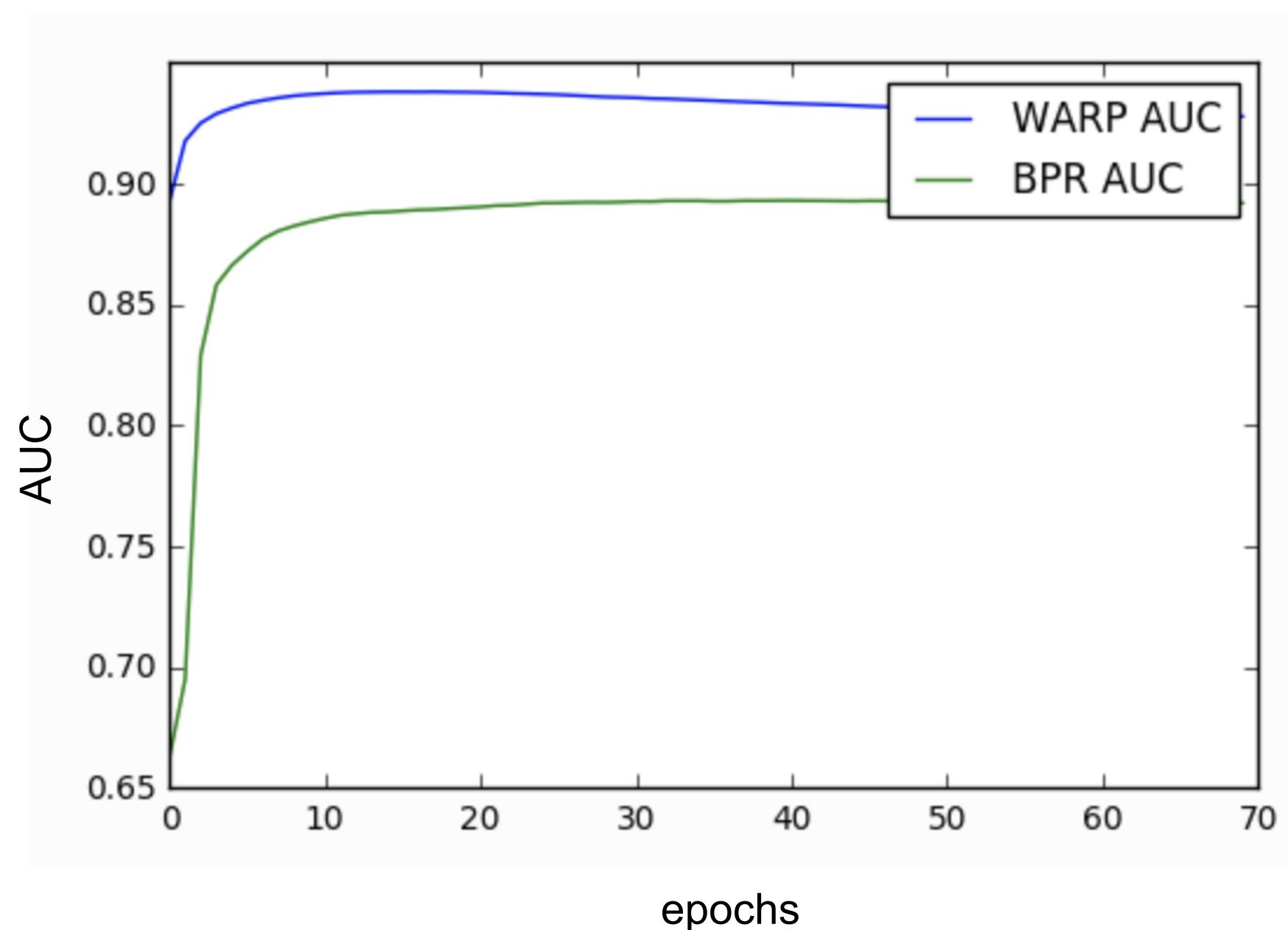
Project weights to enforce constraints, e.g. if  $\|V_i\| > C$  then set  $V_i \leftarrow (CV_i)/\|V_i\|$ .

**end if**

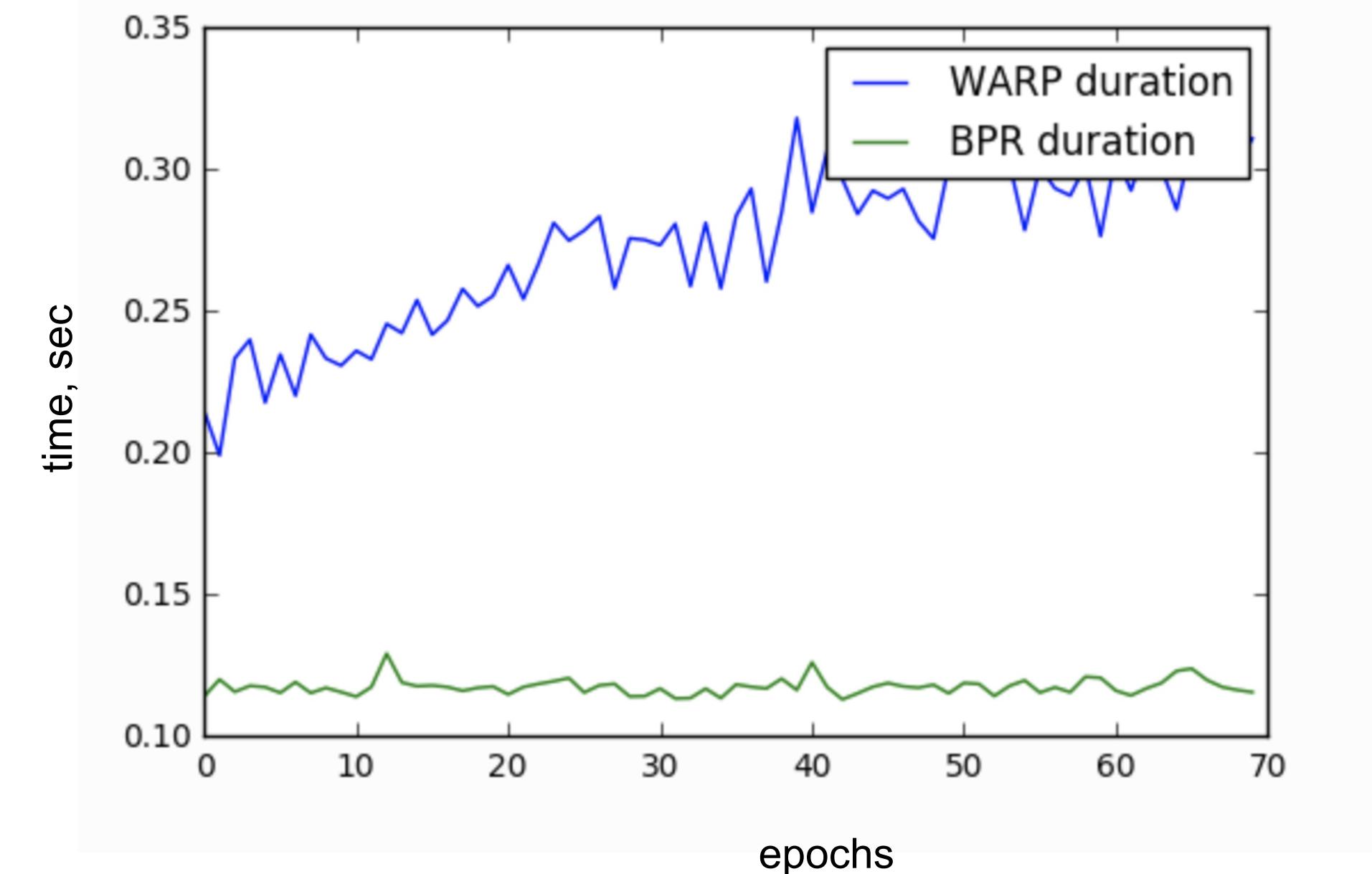
**until** validation error does not improve.

---

# WARP vs. BPR\*



Обычно, WARP лучше по качеству...



но более медленный =(

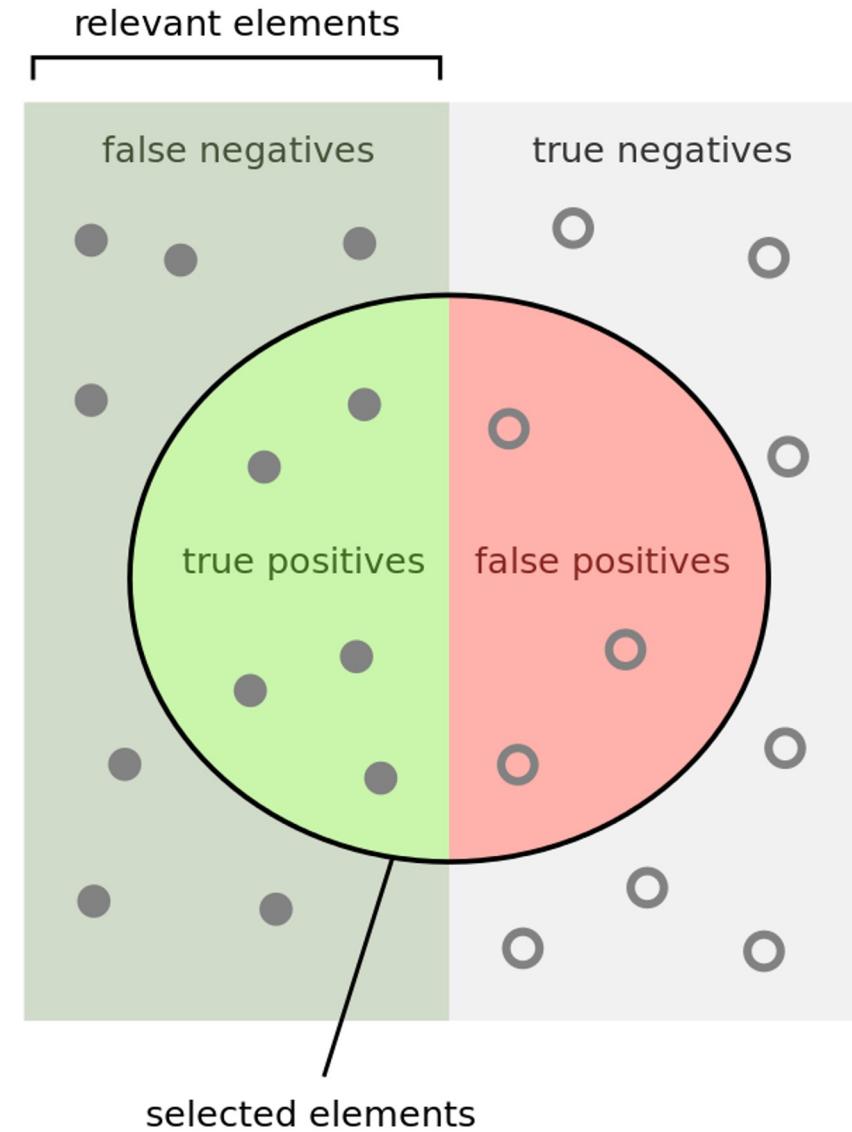
# Метрики ранжирования

# Зоопарк метрик качества ранжирования

---

- Hit Rate (Hit ratio)
  - Precision@k
  - Recall@k
  - AP@k, MAP@k, MNAP@k
  - NDCG@k
  - MRR
- etc.

# Hitrate, Precision@k, Recall@k



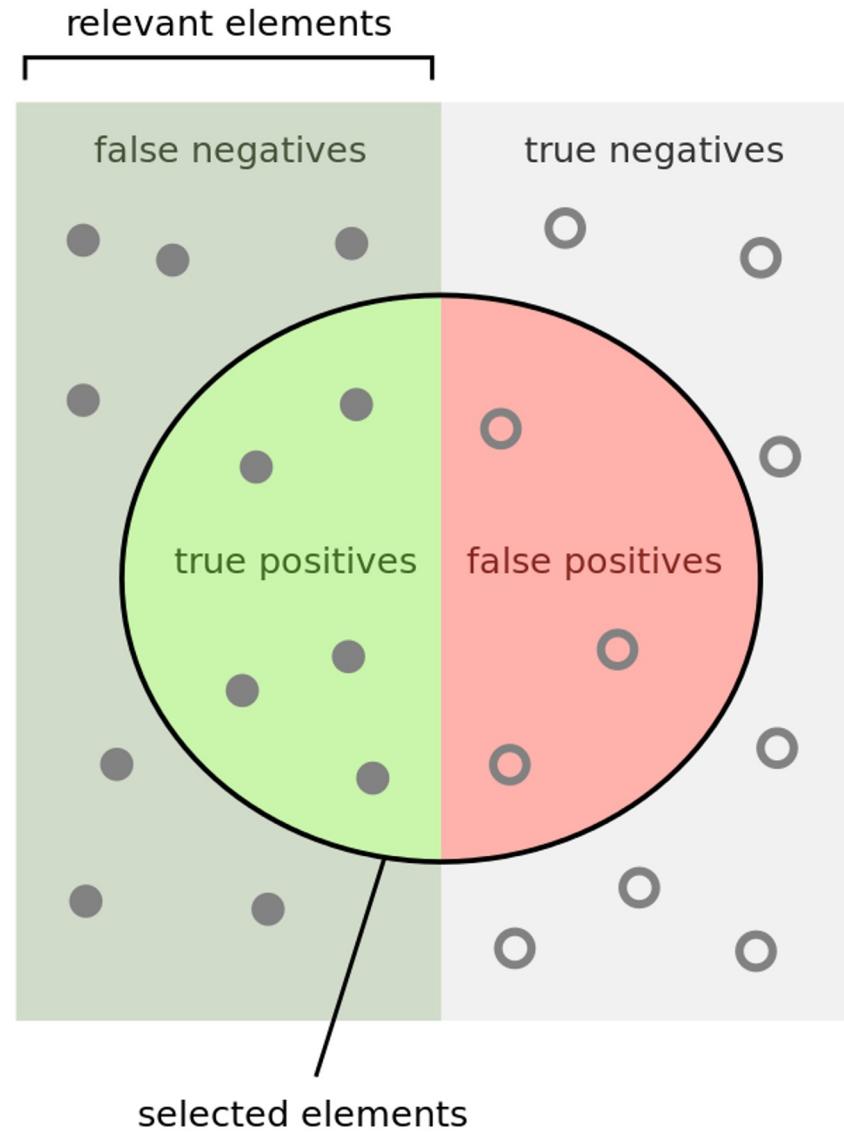
$$\text{Hit Rate} = \frac{\# \text{ hits}}{\# \text{ hits} + \# \text{ misses}}$$

aka Accuracy

How many selected items are relevant?	How many relevant items are selected?
---------------------------------------	---------------------------------------

$$\text{Precision} = \frac{\text{green circle}}{\text{green circle} + \text{red circle}}$$
$$\text{Recall} = \frac{\text{green circle}}{\text{green circle} + \text{grey circle}}$$

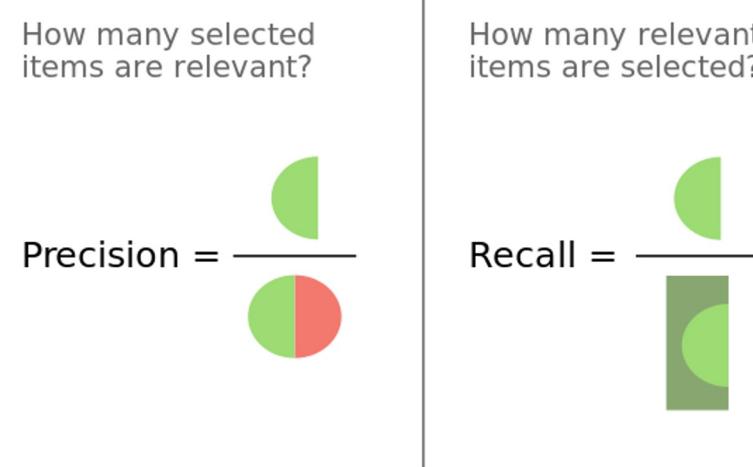
# Hitrate, Precision@k, Recall@k



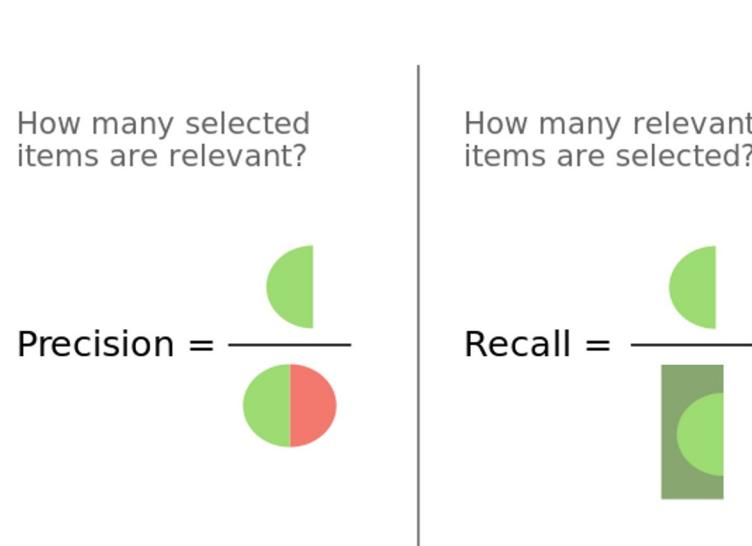
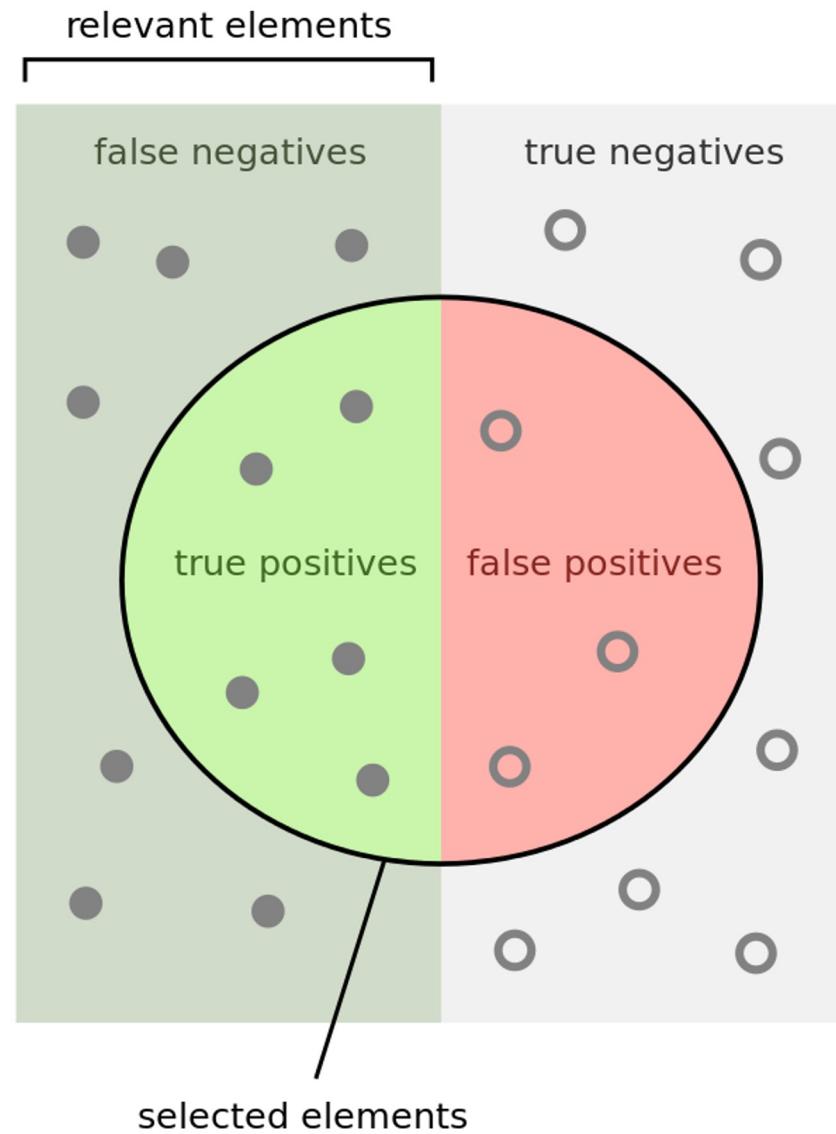
$$Hit\ Rate = \frac{\# \ hits}{\# \ hits + \# \ misses}$$

**Precision – ?**  
**Recall – ?**

Эти метрики зависят от количества рекомендаций для каждого пользователя?



# Hitrate, Precision@k, Recall@k



$$Hit Rate = \frac{\# hits}{\# hits + \# misses}$$

**Precision** – сколько из рекомендованных объектов оказались релевантными.

**Recall** – сколько из релевантных для пользователя объектов мы порекомендовали.

Precision@k, Recall@k – фиксируем для каждого пользователя топ k рекомендованных объектов, считаем метрику по бинарному таргету (0 – нерелевантная рекомендация, 1 – релевантная).

# Посчитаем precision@k, recall@k



- документы выдачи по запросу, белым цветом отмечены нерелевантные

$y_i$  - истинная релевантность айтема на  $i$ -й позиции,  $y \in Y$

$\hat{y}_i$  - предсказанная релевантность

$$\text{precision}@k = \frac{1}{k} \sum_{i=1}^k [\hat{y}_i = 1],$$

где  $k$  - длина списка рекомендаций

$$\text{recall}@k = \frac{\sum_{i=1}^k [\hat{y}_i = 1]}{\sum_{i=1}^N [y_i = 1]} = \frac{\sum_{i=1}^k [\hat{y}_i = 1]}{|Y|},$$

где  $N$  - количество всех айтемов

# Посчитаем precision@k, recall@k



- документы выдачи по запросу, белым цветом отмечены нерелевантные

Пускъ  $k=7$

$$precision@7 = \frac{\# predicted\ relevant}{k\ predicted}$$

$$recall@7 = \frac{\# predicted\ relevant}{\# relevant}$$

## Посчитаем precision@k, recall@k



- документы выдачи по запросу, белым цветом  
отмечены нерелевантные

Пусть  $k = 7$

$$precision@7 = \frac{\# predicted relevant}{k predicted} = \frac{3}{7}$$

$$recall@7 = \frac{\# predicted relevant}{\# relevant} = \frac{3}{5}$$

## Посчитаем precision@k, recall@k



- документы выдачи по запросу, белым цветом  
отмечены нерелевантные

Плюс: + просто интерпретировать  
+ легко считать

Минус: - считаем по 1 пользователю, надо усреднить  
- не учитывает порядок по выборке

Пример: 1 2 3 4 5      ■ - релевантный  
□ - нерелевантный  
 $P@5 = \frac{2}{5}$

1 2 3 4 5  
□ □ □ ■ ■  
 $P@5 = \frac{2}{5}$

# Average precision @ k (AP@k)

Учитывает порядок рекомендованных объектов в топе k.

$$AP@k = \sum_{i=1}^k \frac{y_i}{\sum_{j=1}^k y_j} \cdot p@i = \frac{\sum_{i=1}^k y_i \cdot p@i}{\sum_{i=1}^k y_i}$$

*количество релевантных айтемов*

Пример:

$p@$	1	2	3	4	5	
$y_i$	1	1	0	0	0	total=2

$AP@5 = \frac{1+1+0+0+0}{5} = 0.4$

$p@$	1	2	3	4	5	
$y_i$	0	0	0	1	1	

$AP@5 = \frac{0+0+0+\frac{1}{4}+\frac{2}{5}}{5} = \frac{13}{40} = 0.325$

# Mean average precision (MAP@k)

$$\text{MAP}@k = \frac{1}{|U|} \sum_{u \in U} \text{AP}@k_u,$$

где  $U$ - множество пользователей

Зачем нужен MNAP@k?

$$\text{MNAP}@k = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\min(n_u, k)} \cdot \text{AP}@k_u,$$

где  $n_u$  - кол-во айтемов, с которыми  
пользователь произвзодил взаимодействия

# Normalized Discounted Cumulative Gain (DCG@k)

$$DCG@k = \sum_{i=1}^k g(y_i) d(i) = \sum_{i=1}^k \frac{2^y - 1}{\log(i+1)}$$

g( $y_i$ ) - функция прироста, gain  
d( $i$ ) - регуляризация (штраф), discount

экспоненциальный  
прирост при линейной  
расле перевыполнности

сумма гаинов  
перевыполненной альтернативы  
в вагате, наим  
больше затрач

$$NDCG@k = \frac{DCG@k}{IDCG@k}, \text{ где } IDCG@k - \text{идеальное (max)} \text{ значение DCG@k}$$
$$IDCG@k = \sum_{i=1}^k \frac{1}{\log(i+1)}$$

## Измерение качества рекомендаций

---

**Diversity** (разнообразие) - число рекомендаций из разных категорий, степень различия рекомендаций между сессиями пользователя, различие рекомендаций между пользователями.

**Novelty** (новизна) - сколько среди рекомендаций новых для пользователей объектов.

**Coverage** (покрытие) - доля объектов, которые хотя бы раз побывали среди топа рекомендаций

**Serendipity** (догадливость) - способность угадывать неожиданные нетривиальные предпочтения пользователя.

- Можно выбрать один из них!
- Зачем выбирать, если можно оптимизировать линейную комбинацию из них!

# Рекомендованные источники

---

- [ACM RecSys](#). Tutorials and workshops
- Лекция Евгения Соколова по [задаче ранжирования](#). Лекция Евгения Соколова по [memory-based подходам, коллаборативной фильтрации, MF](#).
- Воронцов [про коллаборативную фильтрацию](#)
- Лекция Сергея Николенко про [подходы для рекомендательных систем](#)
- [ИТМО Wiki. Задача ранжирования](#)
- Библиотека [RecBole](#) с 80+ алгоритмами RecSys
- Библиотека [Microsoft recommenders](#)
- Библиотека [implicit](#) в частности с оптимальной реализацией ALS. Оригинальная статья по [ALS](#)
- Библиотека [LightFM](#) с реализацией этого алгоритма по [статье](#). Это контентная модель - то есть, можно обучать как ALS, а можно еще добавлять признаки по пользователям и объектам.
- Обучение градиентного бустинга для задачи рекомендаций. [Catboost ranking task and metrics](#)