

# Классические алгоритмы рекомендаций

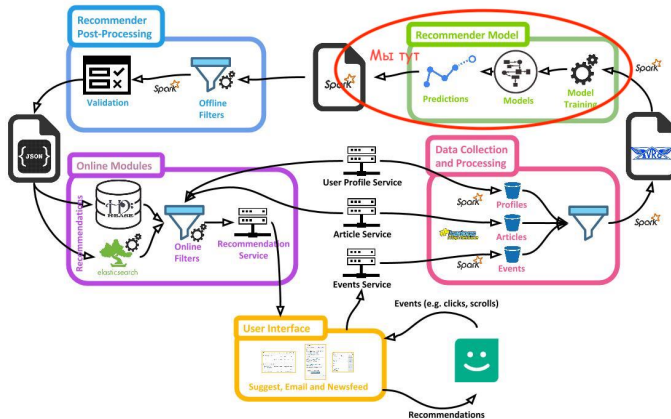
Николай Анохин

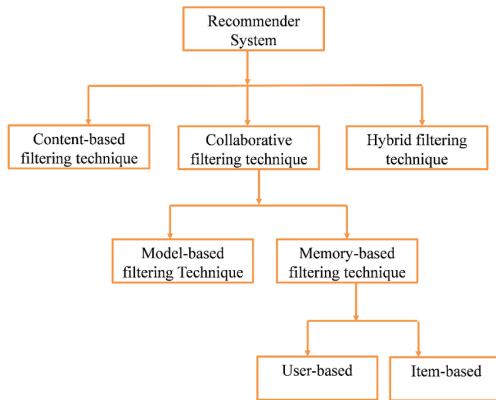
13 октября 2021 г.

## Программа модуля

Дата	Тема	Семинар	Домашка
2021-09-30	Рекомендательные сервисы в продакшене	✓	
2021-10-07	Метрики и базовые подходы	✓	
2021-09-14	Классические алгоритмы рекомендаций	✓	✓
2021-09-21	Нейросетевые рекоммендеры	✓	
2021-09-28	Нерешенные проблемы и новые направления	✓	

# Контекст



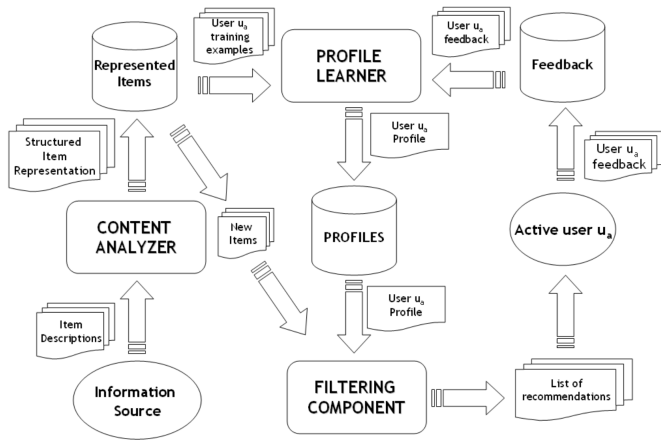


## Content-based RS

## Идея

Рекомендуем пользователю айтемы, похожие на те, что нравились ей раньше

## Архитектура CBRS [RRSK10]



## Анализ контента

Данные	Признаки
Любые Текст	Категориальные / числовые BOW / TF-IDF / BM25 / Эмбед- динги
Картинки	SIFT/SURF/Эмбеддинги
Музыка	Spectral

## Supervised профили пользователей

### Модель

$$p(u \text{ likes } i) = f(x_i, x_u, \theta)$$

$$recommendations = \arg_i \text{ top } k \ p(u \text{ likes } i)$$

Обучаемые параметры:

- $x_u$  – профиль пользователя
- $\theta$  – параметры модели

Данные:

- $\{(x_i, u_j \text{ likes } x_i)\}^N$

Примеры моделей:

- Naive Bayes
- Rocchio
- Meta-learning



## Unsupervised профили пользователей

### Идея

Храним айтемы, с которыми взаимодействовал пользователь, и рекомендуем ближайшие к ним.

Когда айтемов у пользователя слишком много:

- Храним последние
- Кластеризуем и храним представления кластеров [PEZ<sup>+</sup>20]

# Ha Spark

Tokenizer

<https://spark.apache.org/docs/latest/ml-features.html#tokenizer>

TF-IDF

<https://spark.apache.org/docs/latest/ml-features#tf-idf>

Word2Vec

<https://spark.apache.org/docs/latest/ml-features#word2vec>

## Плюсы

- Рекомендации строятся независимо для каждого пользователя
- Рекомендации часто можно объяснить
- Естественная поддержка холодного старта айтемов

## Минусы

- Полагаются на (несовершенные) техники анализа контента
- Отсутствие новизны: умеют рекомендовать только похожие айтемы
- Нет поддержки холодного старта пользователей

## Collaborative filtering<sup>1</sup>-based RS

## Идея

Рекомендуем пользователю айтемы, которые понравились похожим на нее пользователям.

<sup>1</sup>Оскар за худшее название алгоритма

## Neighbourhood CF [LRU14]

### Идея

Пользователи похожи, если они похоже оценивают одни и те же айтемы

	HP1	HP2	HP3	TW	SW1	SW2	SW3
<i>A</i>	4			5	1		
<i>B</i>	5	5	4				
<i>C</i>				2	4	5	
<i>D</i>		3					3

User-based ( $|U| < |I|$ )

$$\hat{r}_{ui} = h^{-1} \left( \frac{\sum_{v \in N_i(u)} w_{uv} h(r_{vi})}{\sum_{v \in N_i(u)} w_{uv}} \right)$$

Item-based ( $|U| > |I|$ )

$$\hat{r}_{ui} = h^{-1} \left( \frac{\sum_{j \in N_u(i)} w_{ij} h(r_{uj})}{\sum_{j \in N_u(i)} w_{ij}} \right)$$

- $N_i(u)$  – соседи пользователя  $u$ , которые оценили айтем  $i$
- $N_u(i)$  – соседи айтема  $i$ , которые оценила пользователь  $u$
- $w_{uv}, w_{ij}$  – веса соседей
- $h$  – функция нормализации

Небольшое количество надежных соседей лучше, чем много ненадежных

Как вычислить веса  $w_{uv}$ ,  $w_{ij}$ ?

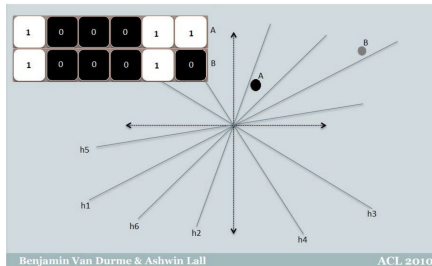
$$\cos(u, v) = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_u} r_{ui}^2 \sum_{i \in I_v} r_{vi}^2}}$$

$$\text{pearson}(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2 \sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}$$



## Locality-Sensitive Hashing для приближенного поиска соседей

The general idea of LSH is to use a family of functions (“LSH families”) to hash data points into buckets, so that the data points which are close to each other are in the same buckets with high probability, while data points that are far away from each other are very likely in different buckets.



## Ha Spark

Bucketed Random Projection for Euclidean Distance

[https://spark.apache.org/docs/2.2.3/ml-features.html#  
bucketed-random-projection-for-euclidean-distance](https://spark.apache.org/docs/2.2.3/ml-features.html#bucketed-random-projection-for-euclidean-distance)

MinHash for Jaccard Distance

[https://spark.apache.org/docs/2.2.3/ml-features.html#  
minhash-for-jaccard-distance](https://spark.apache.org/docs/2.2.3/ml-features.html#minhash-for-jaccard-distance)

## Плюсы

- Простота и интуитивность
- Небольшое количество параметров (1)
- Рекомендации можно объяснить
- Не нужно обучать, удобно добавлять новых пользователей и айтемы

## Минусы

- Разреженность пространства
- User-based: очень пользователей для поиска NN
- Item-based: как понять, для каких айтемов считать рейтинги?

## Model-based CF

### Идея

Выучим модель, которая поможет заполнить “пробелы” в user-item матрице.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
<i>A</i>	4			5	1		
<i>B</i>	5	5	4				
<i>C</i>				2	4	5	
<i>D</i>		3					3

## Бейзлайн

### Модель

$$b_{ui} = \mu + b_u + b_i$$

- $\mu$  – средний рейтинг
- $b_u$  – bias пользователя
- $b_i$  – bias айтема

Оптимизируем

$$\sum_{u,i} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 b_u^2 + \lambda_2 b_i^2 \rightarrow \min_{b_u, b_i}$$

## SVD

## Модель

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

- $q_i$  – латентное представление айтема
- $p_u$  – латентное представление пользователя

Оптимизируем

$$\sum_{u,i} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(b_u^2 + b_i^2 + \|q_i\|^2 + \|p_u\|^2) \rightarrow \min_{b_u, b_i, p_u, q_i}$$

## Как оптимизировать

$$\sum_{u,i} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(b_u^2 + b_i^2 + \|q_i\|^2 + \|p_u\|^2)$$

- ALS [HKV08]
  1. фиксируем  $p_u, b_u$ , оптимизируем  $q_i, b_i$  – получем линрег 1
  2. фиксируем  $q_i, b_i$ , оптимизируем  $p_u, b_u$  – получем линрег 2
  3. повторяем до сходимости
- SGD



## SVD++

## Модель

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left( p_u + \frac{1}{\sqrt{|R(u)|}} \sum_j y_j \right)$$

- $y_j$  – латентное представление айтемов, на которые пользователь дал implicit feedback до оценки айтема  $i$

# Time SVD++

## Модель

$$\hat{r}_{ui} = \mu + b_u(t_{ui}) + b_i(t_{ui}) + q_i^T \left( p_u(t_{ui}) + \frac{1}{\sqrt{|R(u)|}} \sum_j y_j \right)$$

- $t_{ui}$  – время, когда пользователь оценил айтем

## LightFM [Kul15]

## Модель

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$
$$q_i = \sum_{j \in f_i} v_j, \quad p_u = \sum_{j \in f_u} w_j$$

- $f_i$  – признаки айтема
- $v_j$  – латентное представление признаков айтема
- $f_u$  – признаки пользователя
- $w_j$  – латентное представление признаков пользователя

## Альтернативные loss-функции: classification vs regression

Случай implicit feedback похож скорее на задачу классификации, чем регрессии

$$\hat{p}_{ui}(r = 1) = \sigma(\mu + b_u + b_i + q_i^T p_u)$$

Лосс: кросс-энтропия

## Альтернативные loss-функции: ranking with BPR [RFGST09]

Правильное ранжирование важнее, чем точное предсказание рейтинга / фидбэка

$$\hat{p}(u \text{ prefers } i \text{ to } j) = \sigma(\hat{x}_{uij}) = \sigma(\hat{x}_{ui} - \hat{x}_{uj})$$

Лосс:

$$-\sum \log p(u \text{ prefers } i \text{ to } j) + \lambda \|\theta\|^2$$

## Альтернативные loss-функции: WARP [Wil16]

Можно умно семплировать негативные примеры – так, чтобы сложность негативных примеров увеличивалась, когда модель становится точнее.

Дано: пользователь + позитивный айтем

1. Семплируем негативные, пока не найдем неправильно отранжированную пару. Делаем шаг обновления.
2. Чем больше пришлось семплировать, тем меньше learning rate шага обновления.

# Ha Spark

ALS

<https://spark.apache.org/docs/2.2.0/ml-collaborative-filtering.html>

RSVD

<https://github.com/criteo/Spark-RSVD>

## Плюсы

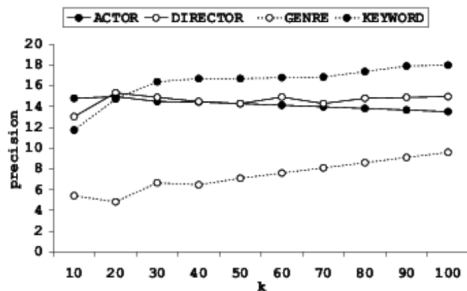
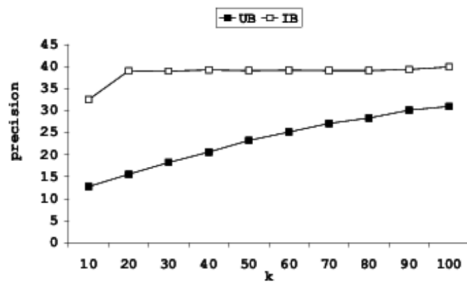
- Качество рекомендаций



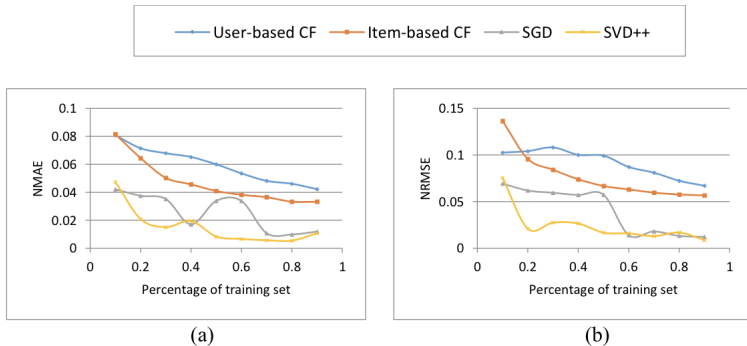
## Минусы

- Сложные алгоритмы оптимизации
- Проблемы с холодным стартом

## CB vs CF



## CF Flavors







**Figure 6.** Performance comparison of different recommendation algorithms at different training ratios

Существуют ситуации, в которых могут пригодиться как СВ, так и CF подходы

Некоторые из них даже реализованы на спарке



## Литература II

-  Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec, *Pinnersage: Multi-modal user embedding framework for recommendations at pinterest*, Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (New York, NY, USA), KDD '20, Association for Computing Machinery, 2020, p. 2311–2320.
-  Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme, *Bpr: Bayesian personalized ranking from implicit feedback*, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (Arlington, Virginia, USA), UAI '09, AUAI Press, 2009, p. 452–461.
-  Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, *Recommender systems handbook*, 1st ed., Springer-Verlag, Berlin, Heidelberg, 2010.
-  Benjamin Wilson, *Warp loss for implicit-feedback recommendation*, Mar 2016.