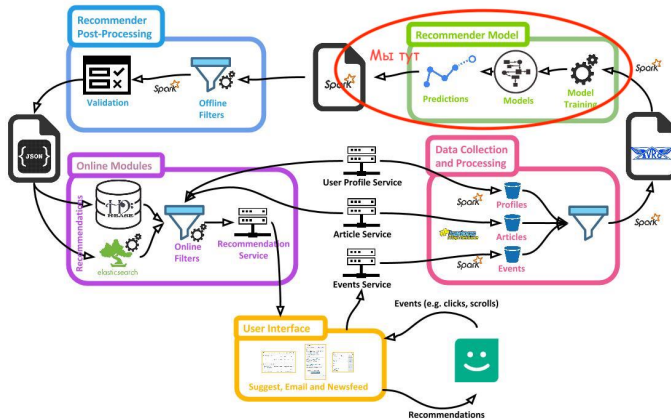


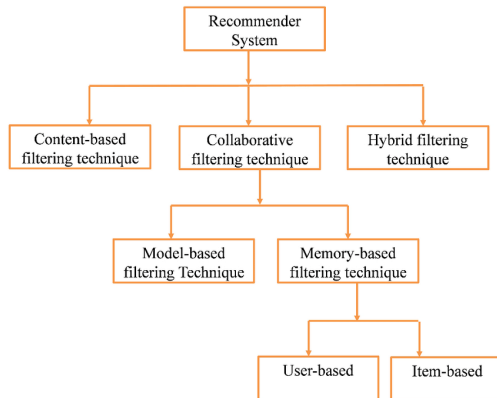
# Классические алгоритмы рекомендаций

Николай Анохин

7 марта 2022 г.

# Контекст





## Content-based RS



## Пример: интуиция

Нравится:

- Возвращение короля
- Король былого и грядущего
- Война мага

Не нравится:

- Новый ум короля

Что порекомендуем?

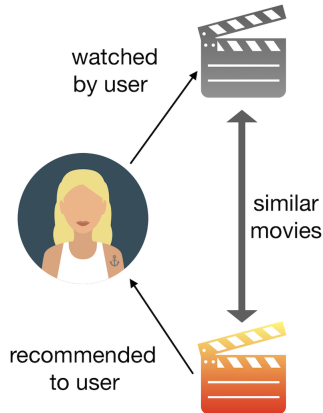
- Битва королей
- Война и мир



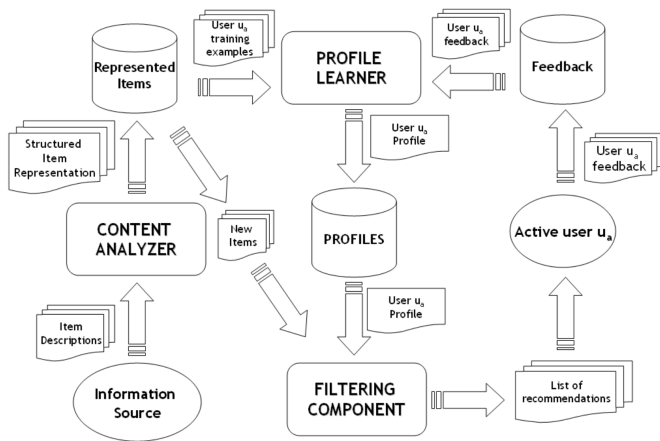
## Content-based RS

### Идея

Рекомендуем пользователю ай-темы, похожие на те, что нравились ей раньше



## Архитектура CBRS [RRSK10]



## Анализ контента

Данные	Признаки
Табличные	





## Анализ контента

Данные	Признаки
Табличные	Категориальные / числовые
Текст	



## Анализ контента

Данные	Признаки
Табличные	Категориальные / числовые
Текст	BOW / TF-IDF / BM25 / NN Эмбединги
Картинки	



## Анализ контента

Данные	Признаки
Табличные	Категориальные / числовые
Текст	BOW / TF-IDF / BM25 / NN Эмбединги
Картинки	SIFT / SURF / NN Эмбединги
Музыка	



## Анализ контента

Данные	Признаки
Табличные	Категориальные / числовые
Текст	BOW / TF-IDF / BM25 / NN Эмбединги
Картинки	SIFT / SURF / NN Эмбединги
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]



## Пример: формально

### Naive Bayes

Нравится:

- Возвращение короля
- Король былого и грядущего
- Война мага

Не нравится:

- Новый ум короля

Что порекомендуем?

1. Битва королей
2. Война и мир

$$p(c|d) \sim p(c)p(d|c) =$$

$$= p(c) \prod_j p(w_j|c) \sim \log p(c) + \sum_j \log p(w_j|c)$$

$$p(w_j|c) = \frac{N_{jc} + \alpha}{N_c + \alpha|V|}$$

Размер словаря  $|V| = 10$ ,  $\alpha = 1$

Вероятности классов  $p(+) = 3/4$ ,  $p(-) = 1/4$

Скоры документов

$$p(+|1) \sim \log 3/4 + \log 1/13 + \log 3/13$$

$$p(-|1) \sim \log 1/4 + \log 1/11 + \log 2/11$$

$$s(1) = p^*(+|1) - p^*(-|1) = 2.69$$



## Известные использования

- Spotify: Deep content-based music recommendation [vdODS13]
- Ozon: Векторное представление товаров Prod2Vec: как мы улучшили матчинг и избавились от кучи эмбедингов [OZO]





## Итоги

### Плюсы

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

### Минусы

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



## Neighbourhood-based Collaborative Filtering



## Пример: интуиция [LRU14]

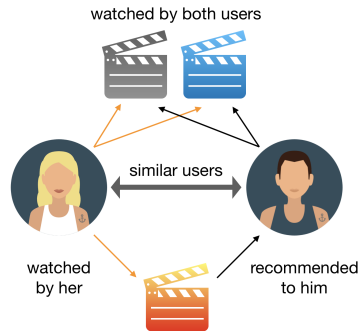
	HP1	HP2	HP3	TW	SW1	SW2	SW3
<i>A</i>	4			5	1		
<i>B</i>	5	5	4	?	?	2	
<i>C</i>				2	4	5	
<i>D</i>		3					3



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

### Идея

Рекомендуем пользователю ай-темы, которые понравились похожим на нее пользователям. Пользователи похожи, если они похоже оценивают одни и те же ай-темы.



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



## User-based

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

## Item-based

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

- $N_i(u)$  – 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



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

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

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



**Дано:**

100 айтеров

1000 пользователей

10000 рейтингов равномерно распределены по пользователям и айтерам

**Вопрос:**

Сколько в среднем общих айтеров у пары пользователей?

Сколько в среднем общих пользователей у пары айтеров?

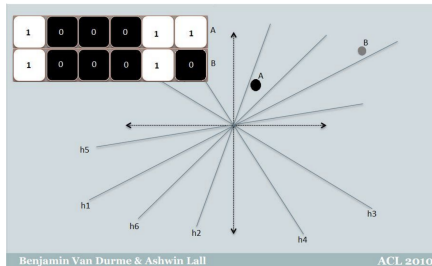
Небольшое количество надежных соседей лучше, чем много ненадежных

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



## 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.





## Пример: формально

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4	?	?	2	
C				2	4	5	
D		3					3

Нормализация:  $h(r) = r$  Веса

$$\cos(A, B) = \frac{4 \cdot 5}{\sqrt{4^2} \sqrt{5^2}} = 0.37$$

$$\cos(B, C) = ?$$

$$r(TW) = ? \quad r(SW1) = ?$$



## Плюсы

- Простота и интуитивность: рекомендации можно объяснить.
- Небольшое количество параметров
- Не нужно обучать, удобно добавлять новых пользователей и айтемы

## Минусы

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



## Model-based Collaborative Filtering



## Model-based CF

### Идея

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

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4	?	?	2	
C				2	4	5	?
D		3					3



## Бейзлайн [KBV09]

Модель

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

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

Оптимизируем

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



## SVD

Модель

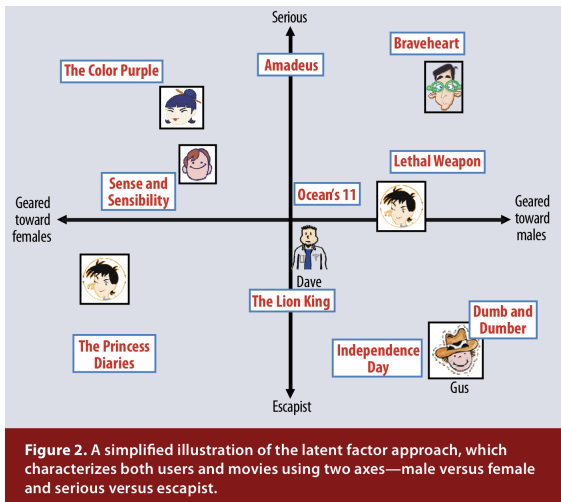
$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

- $q_i$  – 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

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

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



# Time SVD++

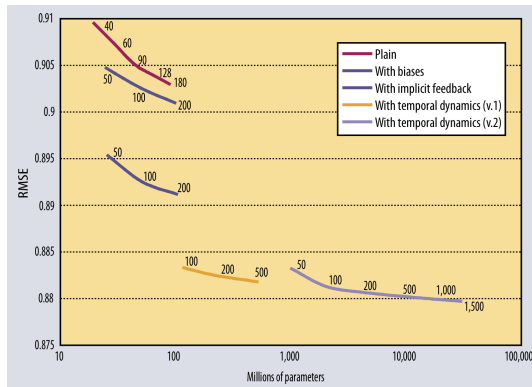
## Модель

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

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



# Netflix Prize



September 21, 2009, the grand prize of US\$1,000,000 was given to the BellKor's Pragmatic Chaos team which bested Netflix's own algorithm for predicting ratings by 10.06



## 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.



## Реализации

	Spark	LightFM	Implicit
SGD		✓	
ALS	✓		✓
RMSE	✓		✓
logistic		✓	✓
BPR		✓	✓
WARP		✓	
GPU			✓





## Плюсы

- Качество рекомендаций: новизна как у neighbourhood-based CF, но можно заполнить все пробелы в user-item матрице
- Большой выбор моделей и лоссов
- Можно переформулировать как нейронную сеть

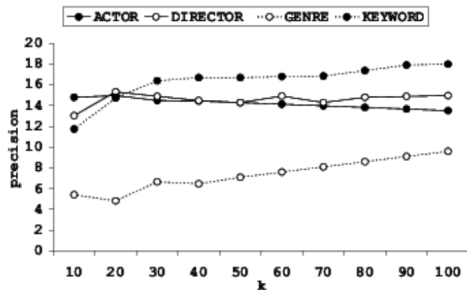
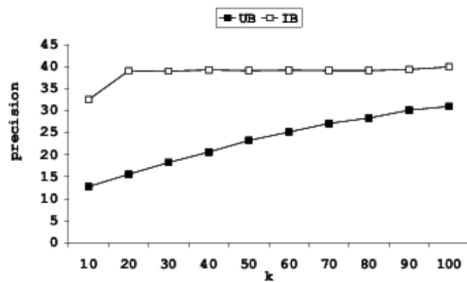
## Минусы

- Сложные алгоритмы оптимизации
- Холодный старт как пользователей, так и айтемов нужно отдельно решать

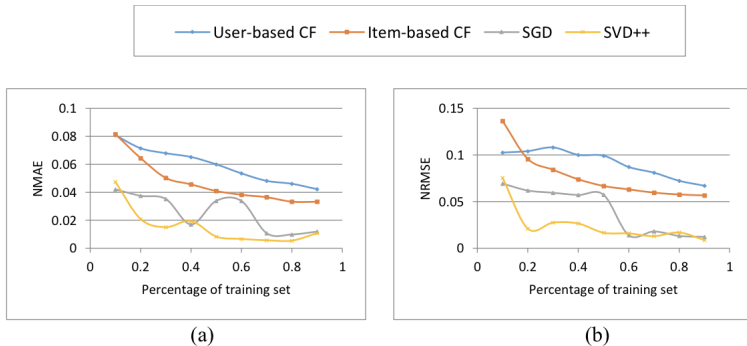


## Итоги

## CB vs CF



## CF Flavors



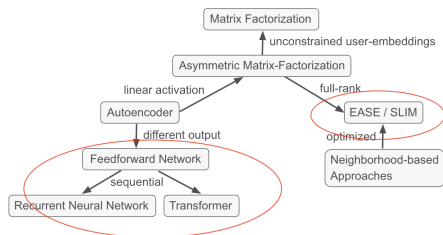
**Figure 6.** Performance comparison of different recommendation algorithms at different training ratios



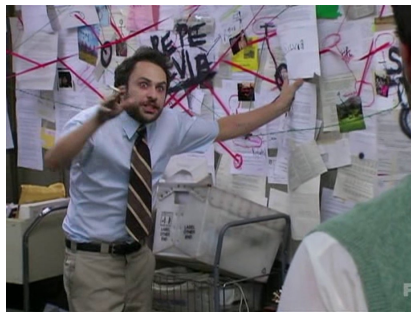
Реальные рекомендеры почти всегда hybrid: применяются как CF, так и CB подходы

“Дефолтным” алгоритмом рекомендаций считается матричная факторизация





## В следующий раз



**FIGURE 2** (Simplified) relationships among several models discussed in this article







## Литература I

-  Yifan Hu, Yehuda Koren, and Chris Volinsky, *Collaborative filtering for implicit feedback datasets*, Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (USA), ICDM '08, IEEE Computer Society, 2008, p. 263–272.
-  Yehuda Koren, Robert Bell, and Chris Volinsky, *Matrix factorization techniques for recommender systems*, Computer **42** (2009), no. 8, 30–37.
-  Maciej Kula, *Metadata embeddings for user and item cold-start recommendations*, Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015. (Toine Bogers and Marijn Koolen, eds.), CEUR Workshop Proceedings, vol. 1448, CEUR-WS.org, 2015, pp. 14–21.
-  Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman, *Mining of massive datasets*, 2nd ed., Cambridge University Press, USA, 2014.



## Литература II

-  Векторное представление товаров *prod2vec*: как мы улучшили матчинг и избавились от кучи эмбедингов.
-  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.





## Литература III

-  Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen, *Deep content-based music recommendation*, Advances in Neural Information Processing Systems (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), vol. 26, Curran Associates, Inc., 2013.
-  Benjamin Wilson, *Warp loss for implicit-feedback recommendation*, Mar 2016.

