<https://habr.com/ru/company/lanit/blog/420499/>

<https://grouplens.org/datasets/movielens/> датасету для тестирования рекомендаций

**Обзор и постановка задачи**

Задача рекомендательной системы – проинформировать пользователя о товаре, который ему может быть наиболее интересен в данный момент времени. Клиент получает информацию, а сервис зарабатывает на предоставлении качественных услуг.

Рекомендательные системы – это про то, что предложить клиенту, чтобы сделать его счастливым.

**Алгоритмы**.  
  
Несмотря на множество существующих алгоритмов, все они сводятся к нескольким базовым подходам, которые будут описаны далее. К наиболее классическим относятся алгоритмы Summary-based (неперсональные), Content-based (модели основанные на описании товара), Collaborative Filtering (коллаборативная фильтрация), Matrix Factorization (методы основанные на матричном разложении) и некоторые другие.

**Неперсонализированные рекомендации**

Начнем с неперсонализированных рекомендаций, поскольку они самые простые в реализации. В них потенциальный интерес пользователя определяется просто средним рейтингом товара: «Всем нравится – значит понравится и вам».

#### Проблема холодного старта

Холодный старт – это типичная ситуация, когда ещё не накоплено достаточное количество данных для корректной работы рекомендательной системы (например, когда товар новый или просто его очень редко покупают). Если средний рейтинг посчитан по оценкам всего трёх пользователей (igor92, xyz\_111 и oleg\_s), такая оценка явно не будет достоверной, и пользователи это понимают. Часто в таких ситуациях рейтинги искусственно корректируют.

#### Актуальность рекомендаций

В некоторых случаях также важно учитывать «свежесть» рекомендации. Это особенно актуально для статей или постов на форумах.

Пример расчета рейтинга в журнале Hacker news:



где U = upvotes, D = downvotes, а P (Penalty) — дополнительная корректировка для имплементации иных бизнес-правил  
  
Расчет рейтинга в Reddit:



где U = число голосов «за», D = число голосов «против», T = время записи. Первое слагаемое оценивает «качество записи», а второе делает поправку на время.

**Content-based рекомендации**

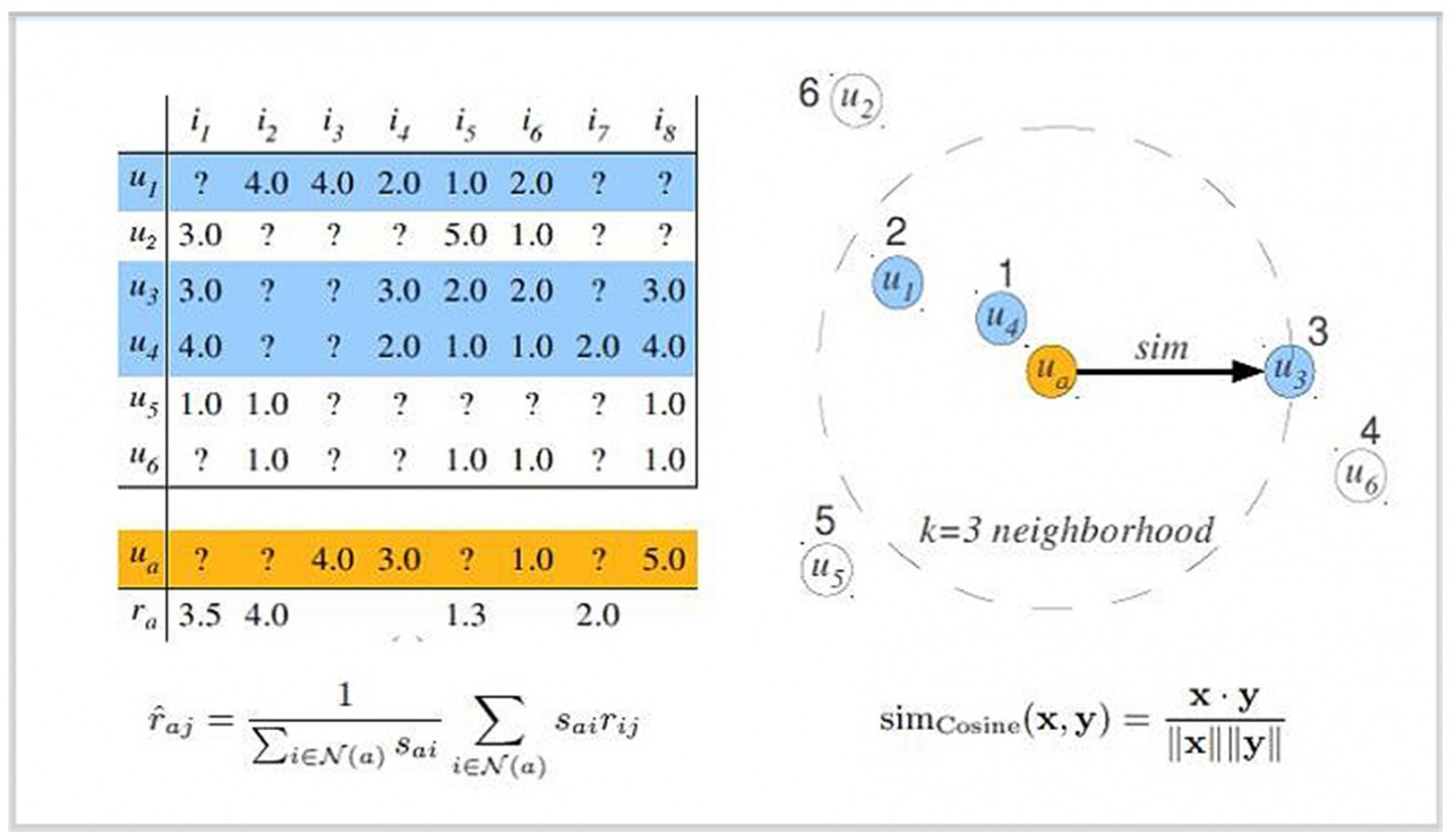
Персональные рекомендации предполагают максимальное использование информации о самом пользователе, в первую очередь о его предыдущих покупках. Одним из первых появился подход content-based filtering.



### Коллаборативная фильтрация (User-based вариант)

В рамках подхода рекомендации генерируются на основании интересов других похожих пользователей. Такие рекомендации являются результатом «коллаборации» множества пользователей.

Классическая реализация алгоритма основана на принципе k ближайших соседей. На пальцах – для каждого пользователя ищем k наиболее похожих на него (в терминах предпочтений) и дополняем информацию о пользователе известными данными по его соседям.



На картинке выше проиллюстрирован принцип работы метода. В матрице предпочтений желтым цветом выделен пользователь, для которого мы хотим определить оценки по новым товарам (знаки вопроса). Синим цветом выделены три его ближайших соседа.

**Стандартизация данных (scaling)**

Поскольку все пользователи оценивают по-разному – кто-то всем подряд пятерки ставит, а от кого-то четверки редко дождешься – перед расчетом данные лучше нормализовать, т.е. привести к единой шкале, чтобы алгоритм мог корректно сравнивать их между собой.

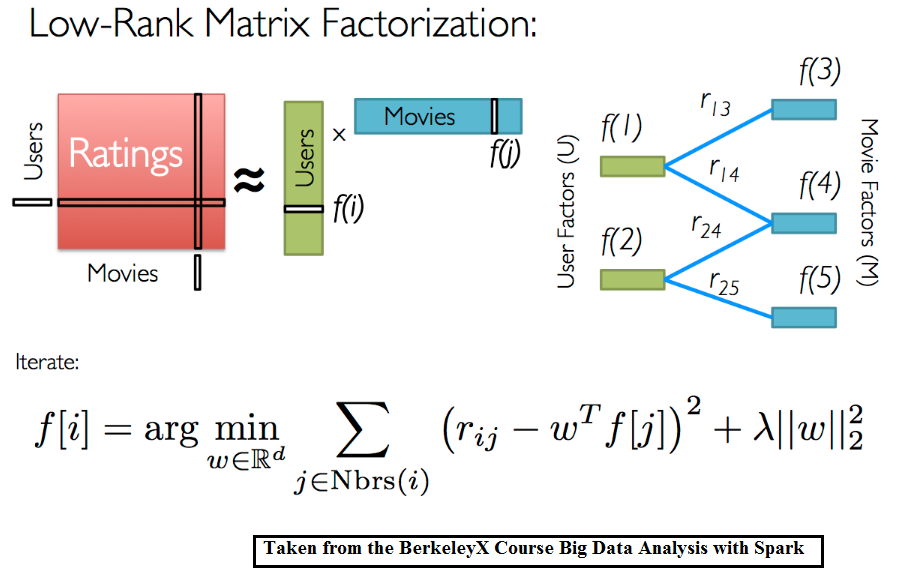
### Коллаборативная фильтрация (Item-based вариант)

Подход Item-based является естественной альтернативой классическому подходу User-based, описанному в первой части, и почти полностью его повторяет, за исключением одного момента — применяется он к транспонированной матрице предпочтений. Т.е. ищет близкие товары, а не пользователей.

### Алгоритмы факторизации

Было бы здорово описать интересы пользователя более «крупными мазками». Не в формате «он любит фильмы X, Y и Z», а в формате «он любит современные российские комедии». Помимо того, что это увеличит обобщаемость модели, это еще решит проблему большой размерности данных — ведь интересы будут описываться не вектором товаров, а существенно меньшим вектором предпочтений.

ALS (alternating least squares) — популярный  итеративный алгоритм разложения матрицы предпочтений на произведение 2 матриц: факторов пользователей (U) и факторов товаров (I). Работает по принципу минимизации среднеквадратичной ошибки на проставленных рейтингах. Оптимизация происходит поочередно, сначала по факторам пользователей, потом по факторам товаров. Также для обхода переобучения к среднеквадратичной ошибке добавляются регуляризационные коэффиценты.



**Другие подходы**

**Ассоциативные правила (Association Rules) -** если мы видим, что молоко в корзину клиент уже положил, самое время напомнить о хлебе.

**RBM (restricted Bolzman Machines) -** ищется наиболее компактное описание пользовательских предпочтений

**Автоэнкодеры (autoencoders) -** получается некий усредненный, очищенный от шума шаблон (данные о пользователе), по которому можно оценить интерес к любому продукту

**DSSM (deep sematic similiarity models) -** в роли латентных переменных здесь внутренние тензорные описания входных данных (embeddings)

### Гибридные решения

Несколько стратегий объединения:

* Weighting — считать средневзвешенный прогноз по нескольким оценкам,
* Stacking — предсказания отдельных моделей являются входами другого (мета)классификатора, который обучается правильно взвешивать промежуточные оценки,
* Switching — для разных продуктов/пользователей применять различные алгоритмы,
* Mixing — вычисляются рекомендации по разным алгоритмам, а потом просто объединяются в один список.

### Резюме

Постановка задачи генерации рекомендаций очень проста — мы составляем матрицу предпочтений с известными нам оценками пользователей, если получается, дополняем эти оценки информацией по клиенту и товару, и пытаемся заполнить неизвестные значения.

[DLRM](https://arxiv.org/pdf/1906.00091.pdf) (Deep-Learning Recommendation Model – рекомендательную модель глубокого обучения

**гибридные модели рекомендации**

## 1. Машина Факторизации (Factorization Machine)

## 2. Широкие и глубокие: нейронная коллаборативная фильтрация (NCF) и Глубокие Машины Факторизации (DeepFM)

## 3. DLRM – рекомендационная модель глубокого обучения – пример <https://github.com/mabeckers/dlrm/blob/new_dataset/Train_DLRM_Digix.ipynb>

**Типы рекомендательных систем**

Существует 4 чипа рекомендательных систем:

* Коллаборативная фильтрация (collaborative filtering).
* Основанные на контенте (content-based).
* Основанные на знаниях (knowledge-based).
* Гибридные (hybrid).

#### Основанные на контенте (content-based)

Товары и услуги рекомендуются на основе знаний о них.

#### Коллаборативная фильтрация (collaborative filtering)

## Рекомендации основаны на истории оценок как самого пользователя, так и других. Во втором случае системы рассматривают потребителей, оценки или интересы которых похожи на ваши.

#### Основанные на знаниях (knowledge-based)

Этот тип работает на основе знаний о какой-то предметной области: о пользователях, товарах и других, которые могут помочь в ранжировании.

#### Гибридные (hybrid)

Есть несколько распространенных типов комбинирования:

* реализация по отдельности коллаборативных и контентных алгоритмов и объединение их предположений;
* включение некоторых контентных правил в коллаборативную методику;
* включение некоторых коллаборативных правил в контентную методику;
* построение общей модели, включающей в себя правила обеих методик.

На просторах интернета мне удалось найти [набор данных](https://grouplens.org/datasets/movielens/) с оценками пользователей различным фильмам.

Коллаборативная фильтрация <https://makesomecode.me/2018/09/movie-recommendation-system/>

content-based

<https://heartbeat.fritz.ai/recommender-systems-with-python-part-i-content-based-filtering-5df4940bd831>

collaborative filtering

<https://heartbeat.fritz.ai/recommender-systems-with-python-part-ii-collaborative-filtering-k-nearest-neighbors-algorithm-c8dcd5fd89b2>

matrix factorization <https://heartbeat.fritz.ai/recommender-systems-with-python-part-iii-collaborative-filtering-singular-value-decomposition-5b5dcb3f242b>