

# Прикладные задачи анализа данных

## Рекомендательные системы

Е. А. Соколов  
ФКН ВШЭ

18 апреля 2021 г.

Задача рекомендательной системы — выбрать для пользователя элементы из некоторого множества, на которых он с наибольшей вероятностью совершит интересные нам действия. Это могут быть:

- товары, которые пользователь захочет купить;
- музыка, которую пользователь захочет дослушать до конца;
- статьи, которые пользователь дочитает до конца;
- видео, которые пользователь досмотрит до конца;
- и многое другое.

Мы будем рассуждать в терминах пользователей (users,  $U$ ) и товаров (items,  $I$ ), но все методы подходят для рекомендаций любых объектов. Будем считать, что для некоторых пар пользователей  $u \in U$  и товаров  $i \in I$  известны оценки  $r_{ui}$ , которые отражают степень заинтересованности пользователя в товаре. Вычисление таких оценок — отдельная тема. Например, в интернет-магазине заинтересованность может складываться из покупок товара и просмотров его страницы, причём покупки должны учитываться с большим весом. В социальной сети заинтересованность в материале может складываться из времени просмотра, кликов и явного отклика (лайки, репосты); это всё тоже должно суммироваться с различными весами. Не будем сейчас останавливаться на этом вопросе, а перейдём к основной задаче.

Требуется по известным рейтингам  $r_{ui}$  научиться строить для каждого пользователя  $u$  набор из  $k$  товаров  $I(u)$ , наиболее подходящих данному пользователю — то есть таких, для которых рейтинг  $r_{ui}$  окажется максимальным.

Получается понятная задача: для объекта  $x_{ui}$  «пользователь-товар» нужно предсказать значение целевой переменной  $r_{ui}$ . Здесь требуют уточнения три пункта: целевая переменная, признаки и функционал ошибки. Первое мы затронули выше, третье обсудим позже, а сейчас поговорим о том, какими признаками можно охарактеризовать пару «пользователь-товар».

# 1 Признаки в рекомендательных системах

## §1.1 Коллаборативная фильтрация

Как понять, что пользователю может понравиться товар? Первый вариант — поискать похожих на него пользователей и посмотреть, что нравится им; также можно поискать товары, похожие на те, которые этот пользователь уже покупал. Методы коллаборативной фильтрации строят рекомендации для пользователя на основе похожестей между пользователями и товарами. Мы рассмотрим два подхода к определению сходства.

### 1.1.1 Memory-based

Два пользователя похожи, если они ставят товарам одинаковые оценки. Рассмотрим двух пользователей  $u$  и  $v$ . Обозначим через  $I_{uv}$  множество товаров  $i$ , для которых известны оценки обоих пользователей:

$$I_{uv} = \{i \in I \mid \exists r_{ui} \ \& \ \exists r_{vi}\}.$$

Тогда сходство двух данных пользователей можно вычислить через корреляцию Пирсона:

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

где  $\bar{r}_u$  и  $\bar{r}_v$  — средние рейтинги пользователей по множеству товаров  $I_{uv}$ .

Чтобы вычислять сходства между товарами  $i$  и  $j$ , введём множество пользователей  $U_{ij}$ , для которых известны рейтинги этих товаров:

$$U_{ij} = \{u \in U \mid \exists r_{ui} \ \& \ \exists r_{uj}\}.$$

Тогда сходство двух данных товаров можно вычислить через корреляцию Пирсона:

$$w_{ij} = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \bar{r}_j)^2}},$$

где  $\bar{r}_i$  и  $\bar{r}_j$  — средние рейтинги товаров по множеству пользователей  $U_{ij}$ . Отметим, что существуют и другие способы вычисления похожестей — например, можно вычислять скалярные произведения между векторами рейтингов двух товаров.

Мы научились вычислять сходства товаров и пользователей — разберём теперь несколько способов определения товаров, которые стоит рекомендовать пользователю  $u_0$ . В подходе на основе сходств пользователей (user-based collaborative filtering) определяется множество  $U(u_0)$  пользователей, похожих на данного:

$$U(u_0) = \{v \in U \mid w_{u_0v} > \alpha\}.$$

После этого для каждого товара вычисляется, как часто он покупался пользователями из  $U(u_0)$ :

$$p_i = \frac{|\{u \in U(u_0) \mid \exists r_{ui}\}|}{|U(u_0)|}.$$

Пользователю рекомендуются  $k$  товаров с наибольшими значениями  $p_i$ . Данный подход позволяет строить рекомендации, если для данного пользователя найдутся похожие. Если же пользователь является нетипичным, то подобрать что-либо не получится.

Также существует подход на основе сходств товаров (item-based collaborative filtering). В нём определяется множество товаров, похожих на те, которые интересовали данного пользователя:

$$I(u_0) = \{i \in I \mid \exists r_{u_0 i_0}, w_{i_0 i} > \alpha\}.$$

Затем для каждого товара из этого множества вычисляется его сходство с пользователем:

$$p_i = \max_{i_0: \exists r_{u_0 i_0}} w_{i_0 i}.$$

Пользователю рекомендуются  $k$  товаров с наибольшими значениями  $p_i$ . Даже если пользователь нетипичный, то данный подход может найти товары, похожие на интересные ему — и для этого необязательно иметь пользователя со схожими интересами.

### 1.1.2 Модели со скрытыми переменными

Все описанные выше подходы требуют хранения разреженной матрицы  $R = \{r_{ui}\}$ , которая может быть достаточно большой. Более того, они весьма эвристичны и зависят от выбора способа вычисления сходства, способа генерации товаро-кандидатов, способа их ранжирования. Альтернативой являются подходы на основе моделей со скрытыми переменными (latent factor models).

Мы будем пытаться построить для каждого пользователя  $u$  и товара  $i$  векторы  $p_u \in \mathbb{R}^d$  и  $q_i \in \mathbb{R}^d$ , которые будут характеризовать «категории интересов». Например, каждую компоненту такого вектора можно интерпретировать как степень принадлежности данного товара к определённой категории или степень заинтересованности данного пользователя в этой категории. Разумеется, никак не будет гарантироваться, что эти компоненты соответствуют каким-то осмысленным категориям, если только мы специально не потребуем этого от модели. По сути, векторы пользователей и товаров являются представлениями (embeddings), позволяющими свести эти сущности в одно векторное пространство.

Сходство пользователя и товара будем вычислять через скалярное произведение их представлений:

$$r_{ui} \approx \langle p_u, q_i \rangle.$$

Также через скалярное произведение можно вычислять сходство двух товаров или двух пользователей.

Мы можем записать функционал ошибки, исходя из способа вычисления сходства:

$$\sum_{(u,i) \in R} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle)^2 \rightarrow \min_{P, Q} \quad (1.1)$$

Суммирование здесь ведётся по всем парам пользователей и товаров, для которых известен рейтинг  $r_{ui}$ . Заметим, что если  $R'$  — матрица  $R$  с центрированными строками и столбцами, то данная задача сводится к низкоранговому матричному разложению:

$$\|R' - P^T Q\|^2 \rightarrow \min_{P, Q}$$

Здесь представления пользователей и товаров записаны в столбцах матриц  $P$  и  $Q$ . Существуют модификации, в которых к скалярным произведениям добавляется масштабирующий множитель  $\alpha \in \mathbb{R}$ :

$$\|R' - \alpha P^T Q\|^2 \rightarrow \min_{P, Q, \alpha}$$

Данный функционал можно регуляризовать:

$$\sum_{(u,i) \in R} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle)^2 + \lambda \sum_{u \in U} \|p_u\|^2 + \mu \sum_{i \in I} \|q_i\|^2 \rightarrow \min_{P, Q} \quad (1.2)$$

Описанная модель носит название Latent Factor Model (LFM).

Отметим, что использование среднеквадратичной ошибки не всегда имеет смысл — в рекомендациях требуется выдать более высокие предсказания для товаров, которые более интересны пользователю, но вовсе не требуется точно предсказывать рейтинги. Впрочем, среднеквадратичную ошибку удобно оптимизировать; более того, именно она использовалась в качестве функционала в конкурсе Netflix Prize, который во многом определил развитие рекомендательных систем и в котором было предложено много популярных сейчас методов.

Существует два основных подхода к решению задачи (1.3). Первый — стохастический градиентный спуск, который на каждом шаге случайно выбирает пару  $(u, i) \in R$ :

$$\begin{aligned} p_{uk} &:= p_{uk} + \eta q_{ik} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle), \\ q_{ik} &:= q_{ik} + \eta p_{uk} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle). \end{aligned}$$

Второй подход основан на особенностях функционала (1.3) и называется ALS (alternating least squares). Можно показать, что этот функционал не является выпуклым в совокупности по  $P$  и  $Q$ , но при это становится выпуклым, если зафиксировать либо  $P$ , либо  $Q$ . Более того, оптимальное значение  $P$  при фиксированном  $Q$  (и наоборот) можно выписать аналитически, — но оно будет содержать обращение матрицы:

$$\begin{aligned} p_u &= \left( \sum_{i: \exists r_{ui}} q_i q_i^T \right)^{-1} \sum_{i: \exists r_{ui}} r_{ui} q_i; \\ q_i &= \left( \sum_{u: \exists r_{ui}} p_u p_u^T \right)^{-1} \sum_{u: \exists r_{ui}} r_{ui} p_u; \end{aligned}$$

(здесь через  $p_u$  и  $q_i$  мы обозначили столбцы матриц  $P$  и  $Q$ ).

Чтобы избежать сложной операции обращения, будем фиксировать всё, кроме одной строки  $p_k$  матрицы  $P$  или одной строки  $q_k$  матрицы  $Q$ . В этом случае можно найти оптимальное значение для  $p_k$  и  $q_k$ :

$$p_k = \frac{q_k(R - \sum_{s \neq k} p_s q_s^T)^T}{q_k q_k^T},$$

$$q_k = \frac{p_k(R - \sum_{s \neq k} p_s q_s^T)}{p_k p_k^T}.$$

Данный подход носит название Hierarchical alternating least squares (HALS) [1].

### 1.1.3 Учёт неявной информации

Выше мы обсуждали, что интерес пользователя к товару может выражаться по-разному. Это может быть как явный (выставление рейтинга или лайк, написание рецензии с оценкой), так и неявный (просмотр видео, посещение страницы) сигнал. Неявным сигналам нельзя доверять слишком сильно — пользователь мог по многим причинам смотреть страницу товара. При этом неявной информации гораздо больше, и поэтому имеет смысл использовать её при обучении моделей.

Один из способов учёта неявной информации предлагается в методе Implicit ALS (iALS) [2]. Введём показатель неявного интереса пользователя к товару:

$$s_{ui} = \begin{cases} 1, & \exists r_{ui}, \\ 0, & \text{иначе.} \end{cases}$$

Здесь мы считаем, что даже если пользователь поставил низкую оценку товару, то это всё равно лучше ситуации, в которой пользователь совсем не поставил оценку. Это не очень сильные рассуждения — пользователь мог просто не найти товар, и в таком случае неправильно судить об отсутствии интереса. Поэтому введём веса  $c_{ui}$ , характеризующие уверенность в показателе интереса  $s_{ui}$ :

$$c_{ui} = 1 + \alpha r_{ui}.$$

Коэффициент  $\alpha$  позволяет регулировать влияние явного рейтинга на уверенность в интересе.

Теперь мы можем задать функционал:

$$\sum_{(u,i) \in D} c_{ui} (s_{ui} - \bar{s}_u - \bar{s}_i - \langle p_u, q_i \rangle)^2 + \lambda \sum_u \|p_u\|^2 + \mu \sum_i \|q_i\|^2 \rightarrow \min_{P, Q}$$

Как и раньше, обучать его можно с помощью стохастического градиентного спуска, ALS или HALS. Предложенные способы вычисления  $s_{ui}$  и  $c_{ui}$  могут изменяться в зависимости от специфики задачи.

### 1.1.4 Факторизационные машины

Рассмотрим признаковое пространство  $\mathbb{R}^d$ . Допустим, что целевая переменная зависит от парных взаимодействий между признаками. В этом случае представляется разумным строить полиномиальную регрессию второго порядка:

$$a(x) = w_0 + \sum_{j=1}^d w_j x_j + \sum_{j_1=1}^d \sum_{j_2=j_1+1}^d w_{j_1 j_2} x_{j_1} x_{j_2}.$$

Данная модель состоит из  $d(d-1)/2 + d + 1$  параметров. Если среди признаков есть категориальные с большим числом категорий (например, идентификатор пользователя), то после их бинарного кодирования число параметров станет слишком большим. Чтобы решить проблему, предположим, что вес взаимодействия признаков  $j_1$  и  $j_2$  может быть аппроксимирован произведением низкоразмерных скрытых векторов  $v_{j_1}$  и  $v_{j_2}$ , характеризующих эти признаки. Мы получим модель, называемую *факторизационной машиной* (factorization machine, FM) [3]:

$$a(x) = w_0 + \sum_{j=1}^d w_j x_j + \sum_{j_1=1}^d \sum_{j_2=j_1+1}^d \langle v_{j_1}, v_{j_2} \rangle x_{j_1} x_{j_2}.$$

Благодаря описанному трюку число параметров снижается до  $dr + d + 1$ , где  $r$  — размерность скрытых векторов.

Данная модель является обобщением моделей с матричными разложениями. Задачу LFM

$$\sum_{(u,i) \in R} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle)^2 \rightarrow \min_{P, Q} \quad (1.3)$$

можно сформулировать как задачу построения регрессии с двумя категориальными признаками: идентификатором пользователя и идентификатором товара. Целевым признаком является рейтинг  $r_{ui}$ . Для некоторого подмножества пар (пользователь, товар) мы знаем рейтинг; для остальных мы хотим его восстановить. После бинаризации признаков получим, что каждый объект  $x$  описывается  $|U| + |I|$  признаками, причём ненулевыми являются ровно два из них: один соответствует номеру пользователя  $u$ , второй — номеру товара  $i$ . Тогда факторизационная машина примет следующий вид:

$$a(x) = w_0 + w_u + w_i + \langle v_u, v_i \rangle.$$

Данная форма полностью соответствует модели (1.3). По сути, факторизационная машина позволяет строить рекомендательные модели на основе большого количества категориальных и вещественных признаков.

Существует несколько методов настройки факторизационных машин, из которых наиболее совершенным считается метод Монте-Карло на основе марковских цепей; реализацию можно найти в библиотеке libFM.

**FFM.** Существует расширение факторизационных машин, позволившее, например, авторам победить в конкурсах Criteo и Avazu по предсказанию кликов по рекламным объявлениям. В обычных факторизационных машинах у каждого признака имеется всего один скрытый вектор, отвечающий за взаимодействие с остальными признаками. Допустим, что признаки можно некоторым образом сгруппировать — например, в задаче рекомендации музыкальных альбомов в бинарном векторе, отвечающем за композиции, будет стоять несколько единиц, соответствующих всем композициям из альбома. Все единицы из этого вектора можно объединить в одну группу. Расширим модель, введя для каждого признака разные скрытые векторы для взаимодействия с разными группами:

$$a(x) = w_0 + \sum_{j=1}^d w_j x_j + \sum_{j_1=1}^d \sum_{j_2=j_1+1}^d \langle v_{j_1, f_{j_2}}, v_{j_2, f_{j_1}} \rangle x_{j_1} x_{j_2},$$

где  $f_{j_1}$  и  $f_{j_2}$  — индексы групп признаков  $x_{j_1}$  и  $x_{j_2}$ . Данная модель носит название *field-aware factorization machines* (FFM) [4].

## §1.2 Контентные модели

В коллаборативной фильтрации используется информация о предпочтении пользователей и об их сходствах, но при этом никак не используются свойства самих пользователей или товаров. При этом может быть полезно находить товары, которые своим описанием похожи на товары из истории пользователя; особенно релевантно это может быть для рекомендательных систем контента (музыки, статей, видео), где пользователю, скажем, захочется познакомиться с музыкой, похожей на музыку его любимых исполнителей.

Как правило, это приводит к следующей идее: все товары описываются с помощью векторов (представлений, *embeddings*), и затем измеряется сходство между вектором нового товара и векторами товаров из истории пользователя. Можно вычислять минимальное или среднее расстояние до векторов из истории. Можно обучить линейную модель, которая для данного пользователя предсказывает целевую переменную на основе представления товара:

$$\sum_{i \in I: \exists r_{ui}} (\langle w_u, q_i \rangle - r_{ui})^2 \rightarrow \min_{w_u},$$

и затем с помощью этой модели оценивать, насколько пользователю подойдут другие товары. Можно обучить граф вычислений, который по всем данным о товаре и о пользователе пытается предсказать целевую переменную. Существует много методов, и какой из них подойдёт для данной задачи — заранее предсказать нельзя.

## §1.3 Статистические признаки

Важны и более простые типы факторов: конверсия просмотра данного товара в покупку за всю историю магазина, число покупок данного пользователя в категории данного товара, число покупок данного пользователя и т.д. Если товар или пользователь уже набрали достаточно статистики, то зачастую такие признаки оказываются самыми главными при принятии решения, поскольку уже содержат в себе достаточно информации о предпочтениях.

# 2 Метрики качества рекомендаций

Существует достаточно много метрик качества рекомендательных систем — некоторые связаны с точностью предсказания, а некоторые оценивают продуктовые аспекты (например, среднее качество тех фильмов, которые обычно рекомендуются). Нет общих советов по поводу того, на какую метрику имеет смысл обращать внимание. Один из возможных подходов — выбрать ключевую с точки зрения бизнеса онлайн-метрику (например, среднее время, которое пользователь проводит на сайте интернет-магазина, средний чек или что-то ещё), а затем выбрать оффлайн-метрику или линейную комбинацию оффлайн-метрик, которая лучше всего коррелирует с ключевой метрикой. Здесь под онлайн-метрикой понимается показатель, который



можно измерить только при запуске рекомендательной системы на реальных пользователях, а под оффлайн-метрикой — функцию, которую можно оценить, построив предсказания модели на исторических данных. Также иногда пытаются найти промежуточную онлайн-метрику, которая коррелирует с основной, но при этом быстрее реагирует на изменения в работе рекомендательной системы — но эту тему мы пока не будем затрагивать. Разберём несколько оффлайн-метрик.

## §2.1 Качество предсказаний

Поскольку рекомендательная система обучается предсказывать оценки  $r_{ui}$ , логично оценивать качество решения именно этой задачи.

**Предсказание рейтингов.** Если модель предсказывает рейтинг или другую вещественную величину (например, длительность просмотра), то качество может измеряться через MSE, RMSE, MAE или другие регрессионные метрики.

**Предсказание событий.** Если модель предсказывает вероятность некоторого события (клика, покупки, просмотра, добавления в корзину), то качество можно измерять с помощью метрик качества классификации — доля правильных ответов, точность, полнота, F-мера, AUC-ROC, AUC-PR, log-loss и т.д.

Также можно учитывать, что мы показываем пользователю только  $k$  товаров, получивших самые высокие предсказания модели, и нас интересует лишь качество этих товаров. Если через  $R_u(k)$  обозначить лучшие  $k$  товаров для пользователя  $u$  с точки зрения модели, а через  $L_u$  товары, для которых действительно произошло интересующее нас событие, то можно ввести следующие метрики:

- Наличие верной рекомендации:  $\text{hitrate}@k = [R_u(k) \cap L_u \neq \emptyset]$ ;
- Точность:  $\text{precision}@k = \frac{|R_u(k) \cap L_u|}{|R_u(k)|}$ ;
- Полнота:  $\text{recall}@k = \frac{|R_u(k) \cap L_u|}{|L_u|}$ .

**Качество ранжирования.** Вообще говоря, нам не очень важно, насколько точно модель предсказывает рейтинг или вероятность клика — от неё лишь требуется дать более релевантным товарам более высокие предсказания. Это значит, что модель должна правильно ранжировать (или сортировать) товары.

Одной из популярных метрик качества ранжирования является nDCG. Обозначим через  $a_{ui}$  предсказание модели для пользователя  $u$  и товара  $i$ . Отсортируем все товары по убыванию предсказания  $a_{ui}$ . Тогда для товара  $i_p$  на позиции  $p$  можно вычислить его полезность  $g(r_{ui_p})$  и штраф за позицию  $d(p)$ . Метрика DCG задаётся как

$$\text{DCG}@k(u) = \sum_{p=1}^k g(r_{ui_p}) d(p).$$

Примерами конкретных функций могут служить  $g(r) = 2^r - 1$  и  $d(p) = \frac{1}{\log(p+1)}$ . Чтобы значение метрики легче было интерпретировать, её можно поделить на



значение DCG при идеальном ранжировании — в этом случае получим метрику nDCG (normalized DCG):

$$\text{nDCG}@k(u) = \frac{\text{DCG}@k(u)}{\max \text{DCG}@k(u)}.$$

Далее значение nDCG можно усреднить по всем пользователям.

**Недостатки оценок качества предсказания.** Основная проблема состоит в том, что качество предсказания само по себе не определяет пользу рекомендательной системы. Модель может идеально угадывать то, что купил пользователь — но, возможно, он приобрёл бы эти товары и без рекомендаций. Поскольку мы никогда не можем узнать, повлияли ли рекомендации на намерения пользователя, имеет смысл анализировать и другие метрики качества, которые могут косвенно говорить о пользе предсказаний модели [5].

## §2.2 Покрытие

**Покрытие товаров.** Полезно обращать внимание на то, какая доля товаров в принципе рекомендуется пользователям — так, может оказаться, что модель показывает только самые популярные товары, а большая часть ассортимента игнорируется. В качестве простейшей метрики можно использовать *покрытие каталога*, которые вычисляются как доля товаров, порекомендованных хотя бы один раз.

Также можно оценить общее разнообразие рекомендаций. Пусть  $p(i)$  — доля показа товара  $i \in I$  среди всех показов для данной рекомендательной системы. Тогда разнообразие можно определить как энтропию такого распределения:

$$H(p) = - \sum_{i \in I} p(i) \log p(i).$$

**Покрытие пользователей.** Рекомендательная система может быть устроена так, что некоторым пользователям вообще ничего не рекомендуется — например, из-за низкой уверенности классификаторов или отсутствия тех или иных признаков для модели. Имеет смысл вычислять долю пользователей, для которых не рекомендуется ни одного товара, чтобы отслеживать проблемы с покрытием в модели рекомендаций.

## §2.3 Новизна

Под новизной понимается доля новых для пользователя товаров среди рекомендованных. При этом под новыми понимаются те товары, которые пользователь видит впервые глобально, а не только на нашем сайте — в идеале хочется уметь угадывать, какие товары пользователь встречал раньше на других ресурсах.

Можно предложить несколько подходов к измерению новизны:

- Для каждого рекомендованного товара добавить в интерфейсе возможность сообщить о том, что этот товар пользователь уже видел.

- Удалить из обучающей выборки часть товаров, которые пользователь купил или просмотрел — тем самым мы будем моделировать ситуацию, в которой пользователь когда-то раньше узнал про этот товар, но в наших данных это не отражено. Далее будем оценивать новизну на основе того, как часто эти удалённые товары попадают в рекомендации.
- Можно считать, что пользователь с большей вероятностью встречал раньше популярные товары и с меньшей — непопулярные. Тогда новизну можно вычислять как долю угаданных рекомендательной системой товаров, где каждый товар имеет вес, обратно пропорциональный популярности этого товара.

## §2.4 Прозорливость (serendipity)

Под прозорливостью понимается способность рекомендательной системы предлагать товары, которые отличаются от всех купленных пользователем ранее. Например, если пользователь читал только книги конкретного автора, то рекомендацию хорошей с точки зрения пользователя книги, но от другого автора, мы будем называть прозорливой.

Прозорливость можно измерять как долю рекомендаций, которые далеки от всех оценённых пользователем товаров. Рассмотрим пример с рекомендациями книг. Допустим, мы хотим измерить расстояние  $d(b, B)$  между новой книгой  $b$  и множеством уже оценённых книг  $B$ . Обозначим через  $c_{B,w}$  число книг от автора  $w$  в множестве  $B$ , а через  $c_B$  — максимальное количество книг от одного автора в  $B$ . Тогда расстояние можно определить как

$$d(b, B) = \frac{1 + c_B - c_{B,w(b)}}{1 + c_B},$$

где  $w(b)$  — автор книги  $b$ .

## §2.5 Разнообразие

Под разнообразием понимается степень сходства товаров внутри одной пачки рекомендаций (т.е. тех товаров, которые одновременно рекомендуются пользователю). Логично ожидать, что полезность набора из 10 чехлов фотоаппаратов ниже, чем набора из чехла, линзы, объектива, батареек и т.д. — именно это и должна оценивать метрика разнообразия. Можно её задавать как, например, среднее попарное расстояние между товарами в одной пачке. Расстояние может измеряться по каталогу (как далеко в дереве категорий товаров находятся эти два товара) или, например, по аналогии с item-to-item рекомендациями (насколько эти два товара пересекаются по множествам купивших их пользователей).

# 3 Архитектура рекомендательных систем

В рекомендательной системе может участвовать очень большое количество товаров. При каждом посещении пользователем веб-страницы, где есть блок рекомендаций, необходимо выдать ему  $k$  наиболее подходящих товаров, причём достаточно

быстро (пользователь не может ждать минуту, пока загрузится страница). В хорошей рекомендательной системе участвуют сотни признаков — их вычисление для каждого товара, а затем ещё и применение ко всем товарам градиентного бустинга или графа вычислений вряд ли получится успеть сделать за 1 секунду. Из-за этого рекомендательные системы работают в несколько этапов: обычно всё начинается с отбора кандидатов, где быстрая модель выбирает небольшое количество (тысячи или десятки тысяч) товаров, а затем только для этих товаров вычисляется полный набор признаков и применяется полноценная модель. В качестве быстрой модели может выступать линейная модель на нескольких самых важных признаках или, например, простая коллаборативная модель.

## Список литературы

- [1] *Gillis, Nicolas and Glineur, François* (2012). Accelerated Multiplicative Updates and Hierarchical Als Algorithms for Nonnegative Matrix Factorization. // *Neural Comput.*, 24, 4, p. 1085–1105.
- [2] *Hu, Yifan and Koren, Yehuda and Volinsky, Chris* (2008). Collaborative Filtering for Implicit Feedback Datasets. // *ICDM '08*.
- [3] *Rendle, S.* (2012). Factorization machines with libFM. // *ACM Trans. Intell. Syst. Technol.* 3, 3, Article 57.
- [4] Field-aware Factorization Machines:  
<http://www.csie.ntu.edu.tw/~r01922136/slides/ffm.pdf>
- [5] *Guy Shani, Asela Gunawardana* (2011). Evaluating recommendation systems. // *Recommender systems handbook*, pp. 257-297. Springer.