Рекомендательные системы (продолжение)

Елена Кантонистова

План

1. Матричные разложения
2. Контентные модели
3. Факторизационные машины
4. ALS для матричной факторизации
5. Метрики
6. Свойства хорошей рекомендательной системы

Обзор Collaborative Filtering

Плюсы:

+ Достаточно неплохие рекомендации при большом количестве явных оценок.

Минусы:

- Два пользователя должны оценивать одинаковые товары, оценка *очень похожих* товаров не учитывается в их близости.
- Проблема холодного старта: не знаем, что делать с новым товаром или пользователем.

Обзор Collaborative Filtering

Минусы:

- Сильная разреженность матрицы оценок приводит к плохим рекомендациям: например, вероятность того, что два пользователя, которые купили 100 книжек каждый, имеют хотя бы одну общую покупку (в каталоге из миллиона книг) равна 0.01 (в случае 50 покупок и 10 миллионов получаем 0.00025).

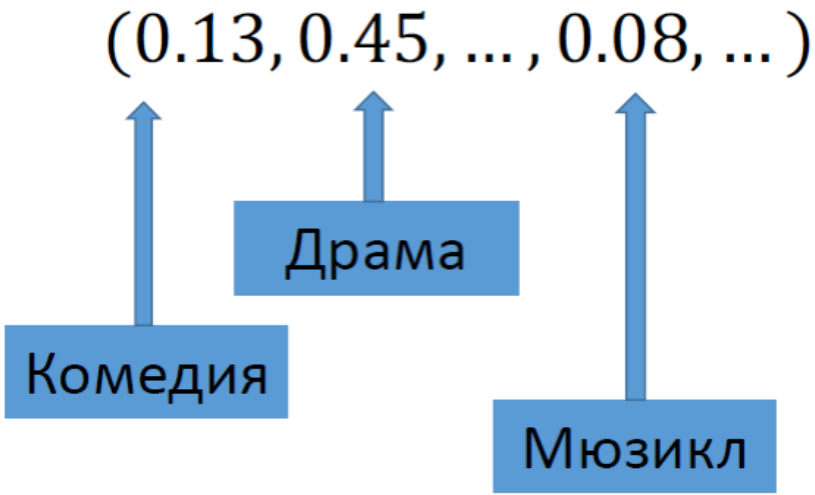
Рекомендации: матричные разложения

Векторы интересов

Решаем задачу рекомендации пользователям различных фильмов.

Можно описать пользователя и фильм векторами интересов:

- для пользователя – насколько он интересуется каждым жанром
- для фильма – насколько он относится к каждому жанру



Рейтинг

Будем определять ***заинтересованность*** как ***скалярное произведение*** вектора пользователя и вектора фильма:

$$(0.1, 0.5, 0.01, 0.92) \times (0, 0, 0.1, 0.95) = 0.875$$

$$(0.1, 0.5, 0.01, 0.92) \times (0.9, 0, 0, 0.1) = 0.182$$

Пользователь

Фильм

Модели со скрытыми переменными

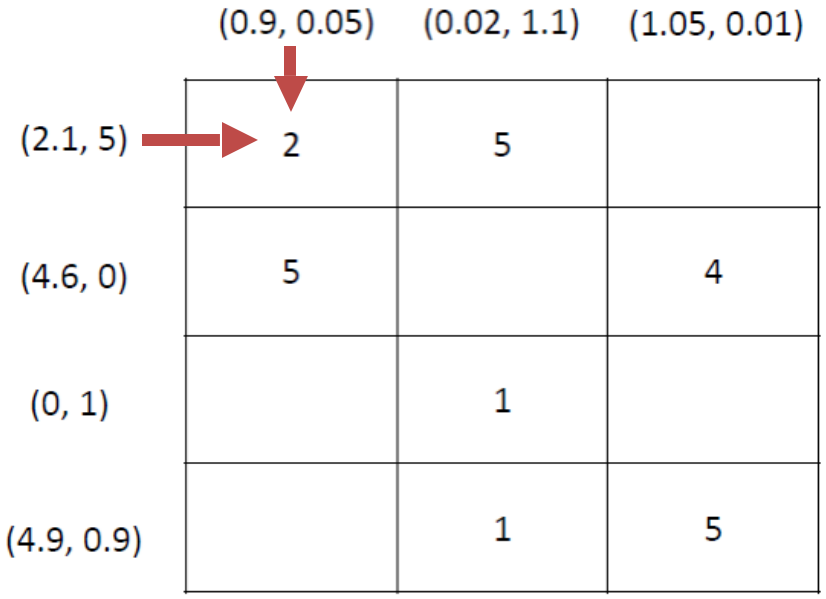
У нас есть матрица рейтингов для задачи пользователь-фильм:

2	5	
5		4
	1	
	2	5

Цель: найти такие векторы пользователей и векторы фильмов, скалярное произведение которых максимально близко к рейтингам из таблицы.

Модели со скрытыми переменными

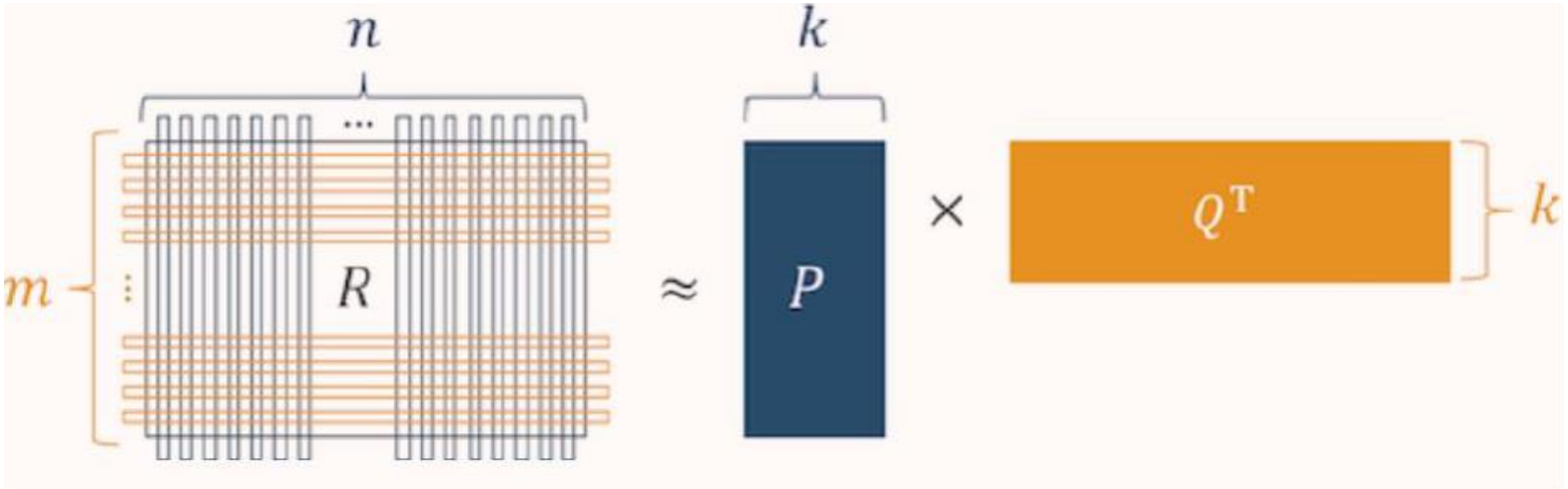
У нас есть матрица рейтингов для задачи пользователь-фильм:



Цель: найти такие векторы пользователей и векторы фильмов, скалярное произведение которых максимально близко к рейтингам из таблицы.

Матричные разложения

Эту задачу можно решить с помощью матричной факторизации, а именно, ***представить матрицу рейтингов как произведение двух матриц***:



- в матрице P находятся векторы интересов пользователей
- в матрице Q находятся векторы фильмов

Singular Value Decomposition (SVD)

Известно, что любая матрица размера $n \times k$ (ранга k) представима в виде $X = VDU^T$, где

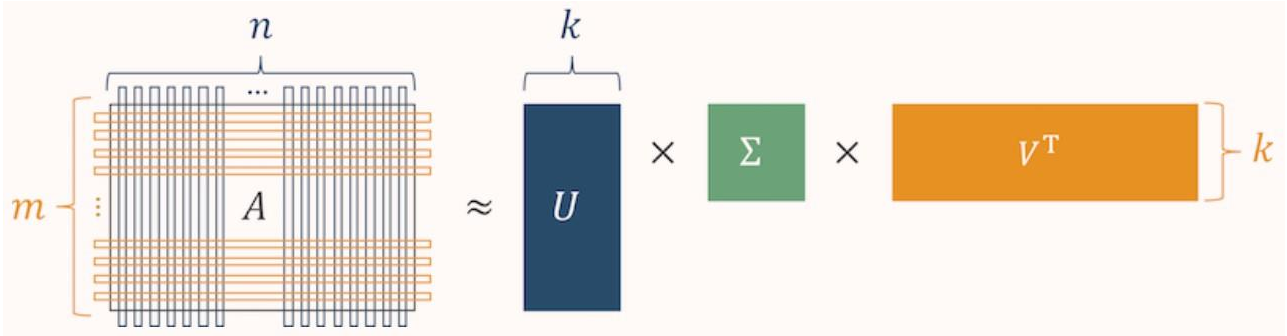
1) V – ортогональная матрица размера $n \times n$, ее столбцы – собственные векторы матрицы XX^T;

2) D – матрица размера $n \times k$, $d_{ii} = \sqrt{\lambda_i}$, $d_{ij} = 0$, если $i \neq j$, где $\{\lambda_i\}_{i=1}^{k}$ – собственные числа матрицы X^TX (и ненулевые собственные значения матрицы XX^T);

3) U – ортогональная матрица размера $k \times k$, её столбцы – собственные векторы матрицы X^TX.

SVD для построения рекомендаций

- Матрица товарных предпочтений (матрица, где строки это пользователи, а столбцы это продукты, с которыми пользователи взаимодействовали) представляется произведением трех матриц:

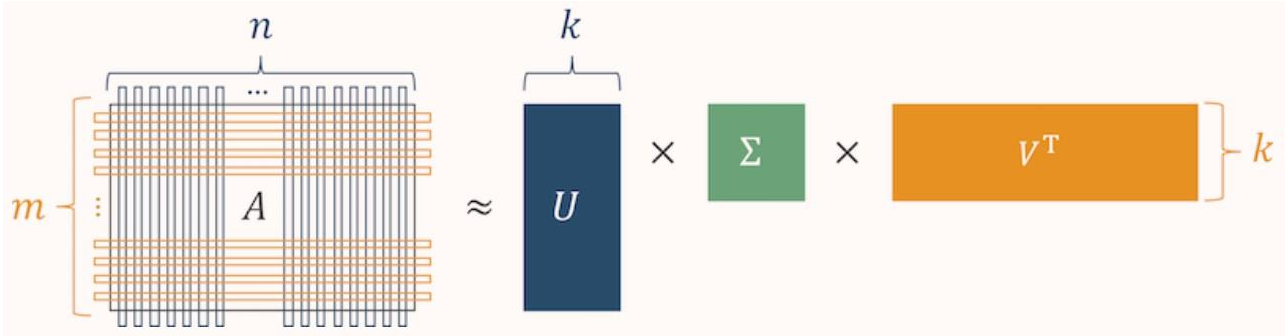


U – описание характеристик пользователя

V – описание характеристик продукта

SVD для построения рекомендаций

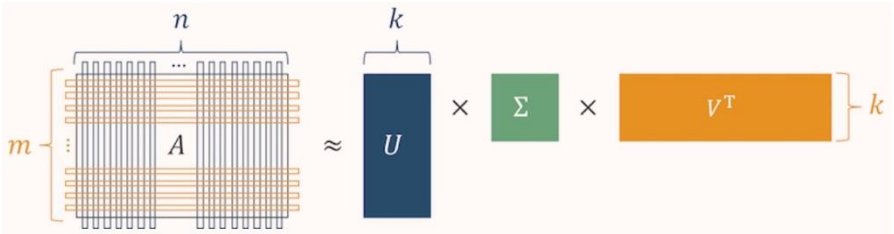
- Матрица товарных предпочтений (матрица, где строки это пользователи, а столбцы это продукты, с которыми пользователи взаимодействовали) представляется произведением трех матриц:



- После восстановления исходной матрицы, клетки, где у пользователя были нули, а появились «большие» числа, показывают степень латентного интереса к товару. Упорядочим эти цифры, и получим список товаров, релевантных для пользователя.

SVD для построения рекомендаций

- Матрица товарных предпочтений (матрица, где строки это пользователи, а столбцы это продукты, с которыми пользователи взаимодействовали) представляется произведением трех матриц:



- После восстановления исходной матрицы, клетки, где у пользователя были нули, а появились «большие» числа, показывают степень латентного интереса к товару. Упорядочим эти цифры, и получим список товаров, релевантных для пользователя.
- При этой операции у пользователя и товара появляются «латентные» признаки. Это признаки, показывающие «скрытое» состояние пользователя и товара.

Общее семейство подобных алгоритмов называется ***NMF (non-negative matrix factorization)***. Как правило вычисление таких разложений весьма трудоемко, поэтому на практике часто прибегают к их приближенным итеративным вариантам.

Контентные модели

Контентные модели

- Теперь мы будем оперировать не матрицей с оценками, а классической для машинного обучения **матрицей объекты-признаки**. Каждый объект будет характеризовать пару user-item и содержать признаки, описывающие как пользователя, так и товар. Кроме этого признаки могут описывать и саму пару целиком

- В этой постановке мы сводим задачу к стандартной задаче машинного обучения. Это может быть **задача классификации или регрессии, в зависимости от целевой переменной**

- Очень часто рекомендации, построенные с помощью этого подхода при помощи известного вам градиентного бустинга, получаются крайне неплохими

Формирование данных для обучения

Формируем строки матрицы объект-признак, где каждая строка соответствует одной паре (*user*,*item*). Ответы для каждой строки - это рейтинг взаимодействия user и item из матрицы рейтингов

Также нам нужны *отрицательные примеры, когда пользователь user не взаимодействует с товаром item*:

- *Рекомендательная система как раз должна выявить те товары, с которыми пользователь еще не взаимодействовал, но взаимодействие было бы/будет успешным*
- Однако большинство пропусков все равно означают отсутствие интереса пользователя к товару. Поэтому возьмем случайные пары *user*,*item*) с пропусками в матрице рейтингов в качестве примеров отрицательного взаимодействия (отсутствия интереса пользователя к товару) и добавим их к обучающей выборке с ответом 0.

Как обучать модель и делать рекомендации

Мы поняли, что каждая строка соответствует паре (*user*,*item*). Заполним строки признаками, которые у нас есть.

Это могут быть:

- характеристики пользователя user
- характеристики товара item
- характеристики пары (user, item)

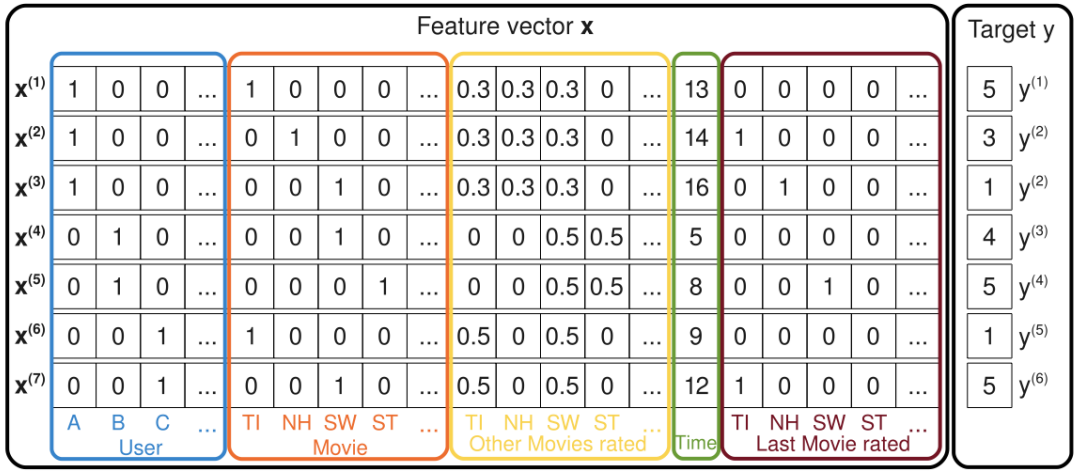
На сформированной выборке **обучаем** выбранную нами модель машинного обучения и делаем прогноз для тестовых (или новых) данных.

Каждому пользователю user **рекомендуем** те товары item, для которых пары (*user*,*item*) получили наиболее высокие прогнозы модели (наибольшие вероятности взаимодействия или наибольший спрогнозированный рейтинг).

Факторизационные машины

Постановка задачи

Factorization machine переформатирует рекомендательную задачу в терминах привычной регрессии (или классификации).
Для каждой пары пользователь-объект создается строчка в обучающем датасете, где пользователь и товар кодируются с помощью one-hot encoding (либо же в матрицу записаны id пользователя и товара, которые модель, например, catboost, закодирует самостоятельно):



Постановка задачи

Целевая переменная, очевидно, зависит от признаков, но также может зависеть от их попарного взаимодействия. Поэтому разумно делать прогноз с помощью квадратичной (по признакам) регрессии, где d – число признаков:

$$a(x) = w_0 + \sum_{j=1}^{d} w_j x_j + \sum_{j_1=1}^{d} \sum_{j_2=j_1+1}^{d} w_{j_1 j_2} x_{j1} x_{j2}$$

Такая модель имеет $d(d-1)/2+d+1$ параметров (весов). В случае если у нас много признаков или же в случае, если в данных есть категориальные признаки с большим количеством категорий, то после one-hot кодирования признаков, а, значит, и весов станет очень и очень много.

Матричная факторизация

Чтобы решить проблему, мы аппроксимируем вес взаимодействия $w_{j_1 j_2}$ скалярным произведением двух векторов меньшей (чем *d*) размерности:

$$w_{j_1 j_2} = (v_{j_1}, v_{j_2})$$

В матричном виде это означает, что мы аппроксимируем матрицу взаимодействий $W = w_{j_1 j_2}$ размера *d*×*d* произведением двух матриц V и V^T размеров *d*×*r* и *r*×*d* соответственно, где *r*<<*d*.

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

Матричная факторизация

Данная модель является обобщением моделей с матричными разложениями.

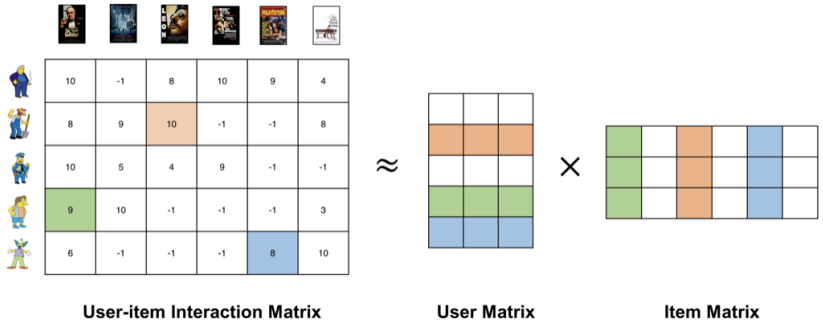
По сути, факторизационная машина позволяет строить рекомендательные модели на основе большого количества категориальных и вещественных признаков.

ALS для матричной факторизации

$$R \approx P^T Q, \ (*)$$

где

- в столбцах матрицы P записаны векторные представления пользователей
- в столбцах матрицы Q записаны векторные представления товаров



Как решить задачу поиска разложения матрицы R?

Важно! Формула (*) верна в случае, если матрица R - матрица с центрированными строками и столбцами (то есть среднее значение по каждой строке и по каждому столбцу равно 0).

В скалярном виде задачу решения уравнения (*) можно записать так:

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

где

- суммирование ведется по всем парам пользователей и товаров, для которых известен рейтинг r_{ui}
- $\overline{r}_u$ - средний рейтинг пользователя u ($\overline{r}_u = 0$, если матрица R имеет центрированные строки)
- $\overline{r}_i$ - средний рейтинг товара i ($\overline{r}_i = 0$, если матрица R имеет центрированные столбцы).

Градиентный спуск

Функционал

$$\sum_{(u,i)\in R}(r_{ui}-\overline{r}_u-\overline{r}_i-(p_u,q_i))^2\to min_{P,Q}$$

также можно регуляризовать:

$$\sum_{(u,i)\in R}(r_{ui}-\overline{r}_u-\overline{r}_i-(p_u,q_i))^2+\lambda\sum_{u\in U}||p_u||^2+\mu\sum_{i\in I}||q_i||^2\to min_{P,Q}.$$

Такая модель носит название ***Latent Factor Model (LFM)***.

Можно использовать и другие виды ошибки (не только квадратичную).

Сложность решения данной задачи оптимизации состоит в том, что мы одновременно должны проводить оптимизацию и по множеству пользователей P, и по множеству товаров Q.

Один из возможных вариантов решения задачи - стохастический градиентный спуск, в котором на каждом шаге случайно выбирается пара $(u,i)\in R$ и по ней обновляются искомые значения векторов:

$$p_{u,k+1}=p_{u,k}+\eta q_{i,k}(r_{ui}-\overline{r}_u-\overline{r}_i-(p_{u,k},q_{i,k})),$$

$$q_{i,k+1}=q_{i,k}+\eta p_{u,k}(r_{ui}-\overline{r}_u-\overline{r}_i-(p_{u,k},q_{i,k})).$$

Здесь k - номер итерации метода и $p_{u,k}, q_{i,k}$ - значения векторов p_u, q_i на k-й итерации.

ALS для решения задачи

Напомним, что нахождения векторов пользователей и товаров мы решаем следующую задачу:

$$\sum_{(u,i)\in R}(r_{ui}-\overline{r}_u-\overline{r}_i-(p_u,q_i))^2+\lambda\sum_{u\in U}||p_u||^2+\mu\sum_{i\in I}||q_i||^2\to min_{P,Q}\ (**)$$

Популярным подходом к решению является метод ***ALS (alternating least squares)***.

Проблема подхода с градиентным спуском заключается в том, что функционал (**) не является выпуклым (в совокупности по P и Q), то есть нет гарантии, что метод не застрянет в локальном минимуме.

Однако, если фиксировать либо P, либо Q, то функционал становится выпуклым! Кроме того, оптимальное значение P при фиксированном Q (и наоборот) можно выписать аналитически:

$$p_u=(\sum_{i:\exists r_{ui}}q_iq_i^T)^{-1}\sum_{i:\exists r_{ui}}r_{ui}q_i,$$

$$q_i=(\sum_{u:\exists r_{ui}}p_up_u^T)^{-1}\sum_{u:\exists r_{ui}}r_{ui}p_u,$$

где p_u и q_i - столбцы матриц P и Q.

HALS для решения задачи

Давайте еще упростим. Зафиксируем все, кроме одной строки p_k матрицы P (или одной строки q_k матрицы Q). Тогда оптимальное значение p_k (q_k) вычисляется по формуле (даже без обращения матриц!):

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

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

Такой подход называется ***Hierarchical alternating least squares (HALS)***.

Метрики

Метрики

- Если предсказываем рейтинг - можно использовать метрики качества регрессии
- Если предсказываем бинарное действие (купит/не купит) - метрики классификации

Обычно показываем пользователю только ***k наиболее релевантных товаров*** $\boldsymbol{R_u(k)}$, так что нужны еще такие метрики:

- $hitrate@k = [R_u(k) \cap L_u \neq 0]$, где $\boldsymbol{L_u}$ **- товары, которые пользователь купит**
- $precision@k = \frac{|R_u(k) \cap L_u|}{|R_u(k)|}$
- $recall@k = \frac{|R_u(k) \cap L_u|}{|L_u|}$

Метрики качества ранжирования (начало)

- Пусть a_{ui} - предсказание модели для пользователя u и товара i
- Отсортируем товары по убыванию предсказаний
- Для товара i_p на позиции p можно посчитать его полезность $g(r_{ui_p})$ и штраф $d(p)$

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

Например, $g(r) = 2^r - 1, d(p) = \frac{1}{log(p+1)}$

Метрики качества ранжирования (начало)

- Пусть $maxDCG@k$ - значение DCG при идеальном ранжировании. Тогда

$$nDCG@k(u) = \frac{DCG@k}{maxDCG@k}$$

Получаем нормализованную метрику, значение которой лежит в отрезке [0; 1].

Свойства хорошей рекомендательной системы

Свойства хорошей рекомендательной системы

Какими свойствами должна обладать хорошая рекомендательная система?

Покрытие (Coverage)

Покрытие товаров. Может так случиться, что модель показывает пользователям только самые популярные товары (или какую-то небольшую группу товаров), а большая часть ассортимента игнорируется. Поэтому полезно смотреть на то, *какая доля товаров из всего каталога рекомендуется пользователям*.

Важно следить за *общей разнообразностью рекомендаций* модели в целом (а не для конкретного пользователя). Ее можно посчитать, например, по формуле энтропии:

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

где $p(i)$ - доля показа товара $i \in I$ среди всех показов для данной рекомендательной системы.

Покрытие (Coverage)

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

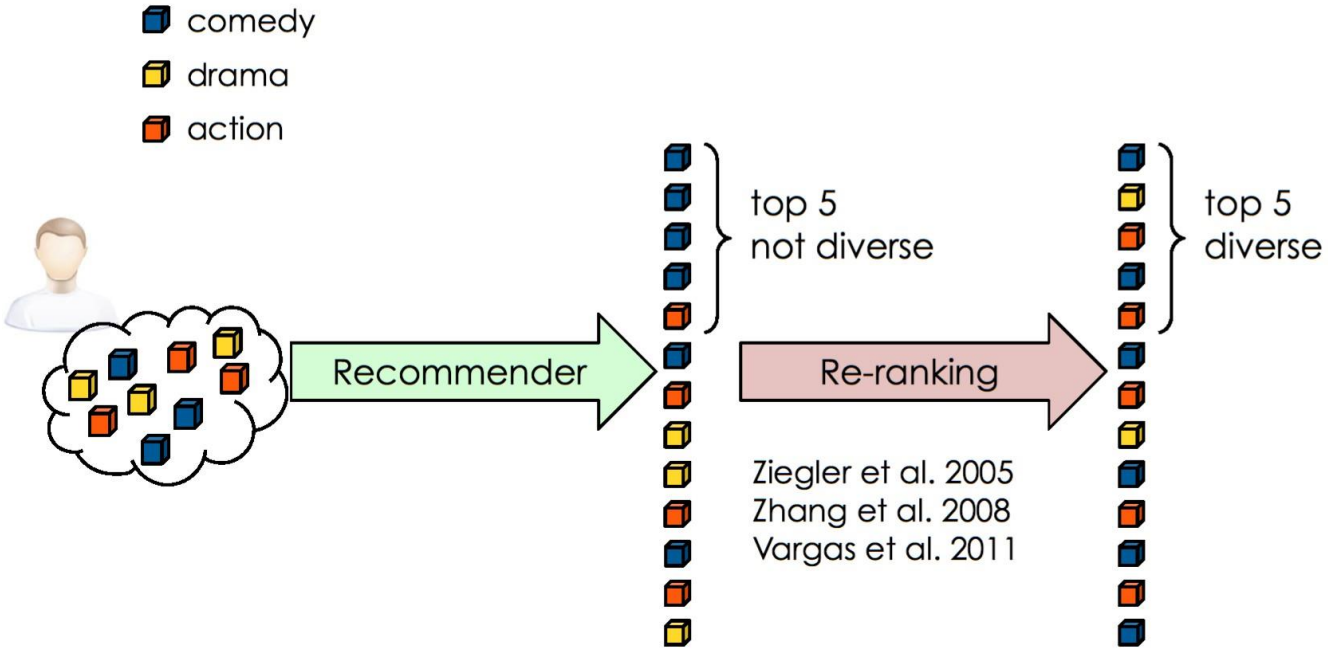
Diversity (разнообразие)

						
Born This Way	*Pink Friday*	*Dangerously in Love*	*Born This Way – The Remix*	*Femme Fatale*	*Can't be Tamed*	*Teenage Dream*
Lady Gaga	Nicki Minaj	Beyoncé	Lady Gaga	Britney Spears	Miley Cyrus	Katy Perry

Разные артисты

Diversity (разнообразие)

Как можно сделать



Novelty (новизна рекомендаций)

- Как оценить?
- Как улучшить?

Serendipity (способность удивить)

Порекомендовать Star Wars II тому, кто посмотрел Star Wars I — очевидно и бесполезно

- Как оценить?
- Как улучшить?

Serendipity (способность удивить)

- Не рекомендовать слишком очевидные фильмы (все друзья это купили/посмотрели)
- Рекомендовать фильмы, которые максимально не похожи на предпочтения пользователя

Serendipity (способность удивить)

Прозорливость можно измерить как **долю рекомендаций, которые далеки от всех оцененных пользователем товаров**.

Пусть, например, мы хотим измерить расстояние $d(b,B)$ между новым фильмом b и множеством уже оцененных пользователем фильмов B.

Пусть

- $c_{B,w}$ - число фильмов жанра w в множестве B
- c_B - максимальное число фильмов одного жанра в множестве B.

Тогда расстояние $d(b,B)$ можно вычислить по формуле

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

где $w(b)$ - жанр фильма b.