Разработка рекомендательной системы на основе текущей сессии пользователя с использованием многоуровневого отбора кандидатов

Воробьева А.С., Мохов А.И.

Консультант: Кислинский В.Г.

Москва, 2020

Классические рекомендательные системы

Рекомендательная система — комплекс алгоритмов, программ и сервисов, задача которого предсказать, что может заинтересовать того или иного пользователя. В основе системы лежит информация о профиле человека и иные данные.

Постановка задачи

По сессии необходимо порекомендовать товары из интернет-магазина.
Пример сессий:

session	view	cart	order
0	id1, id2, id3	id2, id5, id6	id2, id7
1	id3, id6, id9	id11, id2, id6	id3, id1
2	id1, id2, id9	id11, id2, id6	id3, id1

Важный аспект задачи - известны лишь категории товаров, просмотры, добавления в корзину и заказы для конкретной сессии. **Нет привязки к пользователю!**

Примеры рекомендаций для просмотров

Рис.: Просмотренный товар

В карточке товара находятся рекомендации, которые система автоматически сочла подходящими к текущему товару.

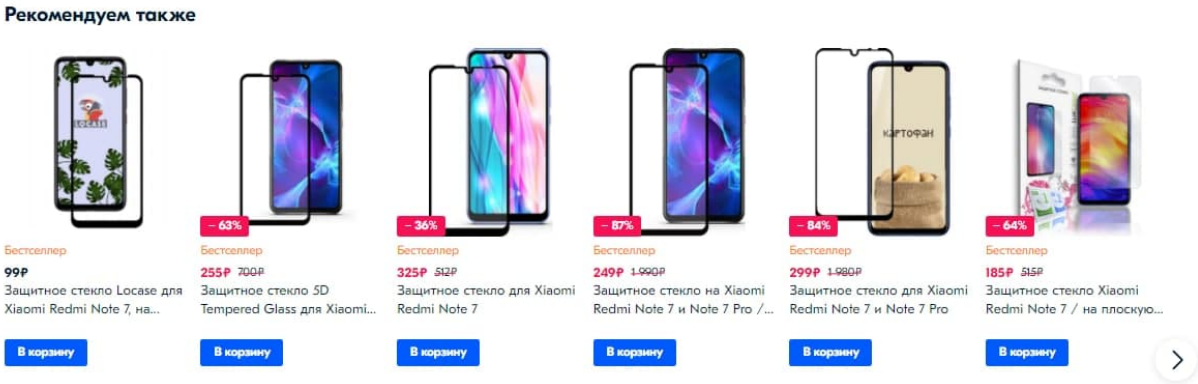


Рис.: Рекомендации на основе просмотров

Примеры рекомендаций для товаров в корзине

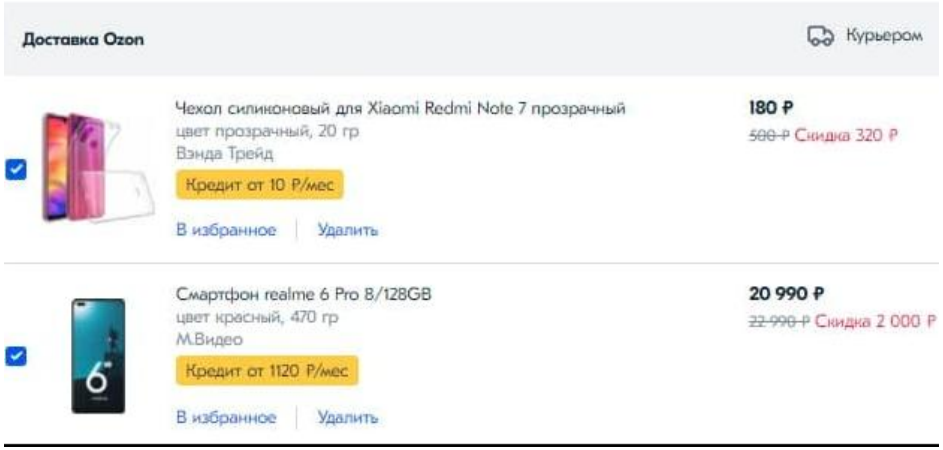


Рис.: Товары в корзине

После перехода в корзину система также формирует и показывает рекомендации, на основе товаров, которые лежат в корзине. Показывая релевантные для пользователя товары можно увеличить средний чек.

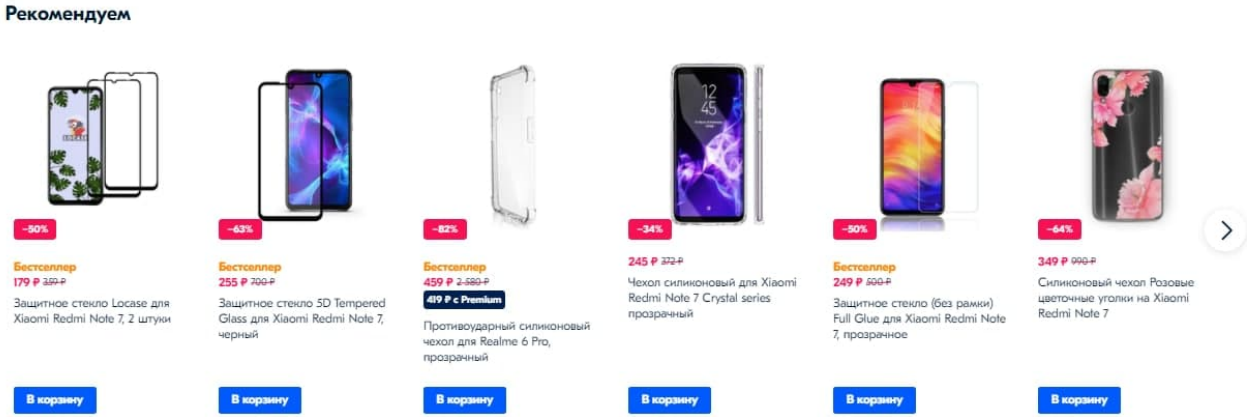


Рис.: Рекомендации на основе текущей корзины

Целевая функция. mAP@K

Precision at K (P@K) - доля рекомендованных объектов, которые входят в K первых релевантных объектов.
Есть список рекомендация r длины N и задано $K \leq N$. Есть истинный список рекомендаций Y, и первые K его элементов обозначим Y_k.
Тогда можно посчитать P@k по формуле:

$$P@K = \frac{\sum_{i=1}^{K} \mathbb{1}_{Y_k}(r_i)}{K}$$

$$\mathbb{1}_{Y_k}(r_i) = \begin{cases} 1, r_i \in Y_k \\ 0, r_i \notin Y_k \end{cases}$$

Average precision at K (AP@K):

$$AP@K = \frac{\sum_{i=1}^{K} P@i}{K}$$

Целевая метрика - Average Precision @ K (AP@k):

$$Q(\hat{y^i}, y^i) = AP@K = \frac{1}{l} \sum_{i=1}^{l} AP@K^i \tag{1}$$

Усредняя данную метрику по всем сессиям, получаем Mean AP@k.

Математическая постановка задачи

Введём отображение $f : id \to X$ - представление id в виде вектора в n-мерном пространства.
Пусть матрицы X^i_{view} и X^i_{cart} - матричные представления сессии под номером i.
Элементами матрицы являются вектора x^i_{view} и x^i_{cart} соответственно, которые представляют уникальные товары в данной сессии, получаемые из отображения f.
Паре (X^i_{view}, X^i_{cart}) соответствует вектор y^i, состоящий из id товаров, которые находятся в **заказе**.
Необходимо найти:

$$\hat{y^i} = A(X^i_{view}, X^i_{cart}) : Q(\hat{y^i}, y^i) \to max \tag{2}$$

То есть найти такое преобразование A над исходными матрицами, что целевая метрика mAP@k достигала максимума.

Как мы решаем задачу

В качестве решения предлагается использовать двухуровневый подход:

1. Отбор кандидатов с помощью простых моделей:
   - Обучение векторных представлений товаров (Word2Vec)
   - Матрица отношений
   - Использование популярных товаров в качестве кандидатов
2. Ранжирование с использованием градиентного бустинга [1, 2].

В результате получаем список рекомендаций для сессии.

Word2Vec - группа языковых моделей на нейронных сетях. Модели позволяют получить векторное представление (embedding) для слов (токенов)[3].
Пусть $w \in W$ - слово, в строковом формате. Введём отображение $f : W \to V$, переводящее множество W в V, где $V \in \mathbb{R}^n$.
Тогда в пространстве $\mathbb{R}^n(\mathbb{E}^n)$ определена **метрика** d, задающая расстояние между любой парой $(x, y)(x \in \mathbb{R}^n, y \in \mathbb{R}^n)$

$$d(x, y) = ||x - y|| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3}$$

Так же определено косинусное сходство между двумя элементами пространства $\mathbb{R}^n$:

$$sim(x, y) = \frac{(x, y)}{\|x\|\|y\|} = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{4}$$

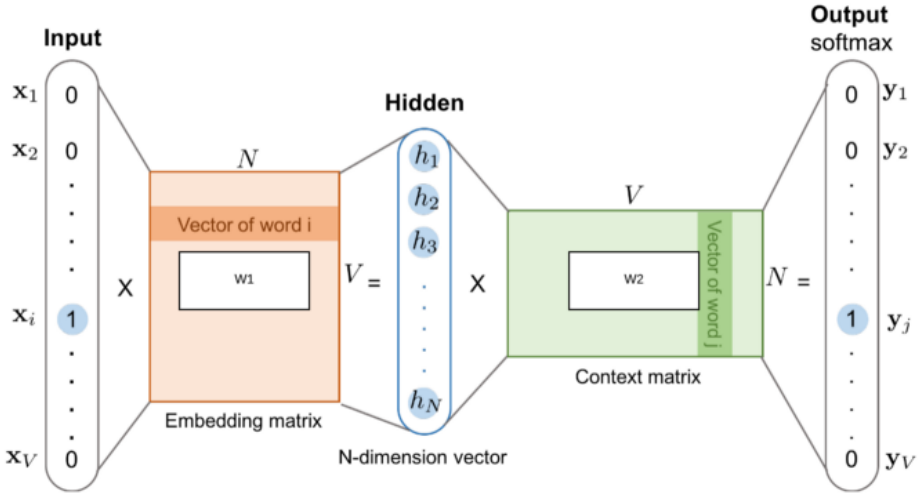
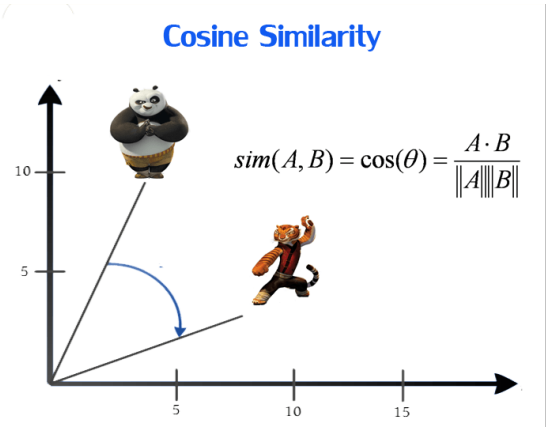
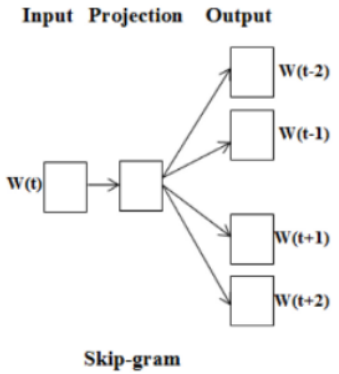


Рис.: Архитектура word2vec





Дано множество сессий S, пусть $S_i, S_j (i \neq j)$ две сессии из этого множества, состоящие из некого набора id товаров. Составим для сессий матрицу C таким образом, что на пересечении индексов k, l будет 1, если товары встречаются в одной сессии и 0, если не встречаются.
Пример:
$S_i = [1, 2, 3], S_j = [3, 4, 5]$. Число уникальных элементов 5. Матрица имеет размерность $5x5$

	1	2	3	4	5
1	0	1	1	0	0
2	1	0	1	0	0
3	1	1	0	1	1
4	0	0	1	0	1
5	0	0	1	1	0

Выберем товары 2 и 3, посчитаем сходство между ними.

$$sim(c_2, c_3) = \frac{(c_2, c_3)}{\|c_2\|\|c_3\|} = \frac{1}{2\sqrt{2}}. \quad (5)$$

Таким образом составляется и общая матрица отношений.

Количество уникальных id товаров $> 2 * 10^6$.
При создании dense матрицы неизбежно возникает ошибка, которая говорит о недостатке памяти.

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
<ipython-input-1-ba941355a018> in <module>
      2
      3 size = 1_000_000
----> 4 np.zeros((size, size))

MemoryError: Unable to allocate 7.28 TiB for an array with shape (1000000, 1000000) and data type float64
```

Нет необходимости хранить нули - их очень много и они тоже место, поэтому используются sparse матрицы. Более того, можно хранить не матрицу отношений, а уже посчитанные значения косинусной схожести в матрице такого же размера и формата. Это избавляет от ненужных вычислений на этапе первичного отбора кандидатов.

```
1 from sklearn.metrics.pairwise import cosine_similarity as sim
2 similarities_view = sim(view.transpose(), dense_output=False)
3 similarities_cart_add = sim(cart.transpose(), dense_output=False)
```

view и *cart* - бинарные матрица отношений для просмотров и добавлений в корзину соотвесвенно.

Получение кандидатов

Используем word2vec и матрицу схожестей для получения кандидатов (*первичные рекомендации*), а так же добавляем к ним список 50 самых популярных товаров из списка заказов.
После получения рекомендации для каждой сессии, таблица будет выглядить следующим образом:

view	cart	order	prediction
$[id_{v1}{}^{1}, ..., id_{vi}{}^{1}]$	$[id_{c1}{}^{1}, ..., id_{cj}{}^{1}]$	$[id_{o1}{}^{1}, ..., id_{ok}{}^{1}]$	$[p_1^1, ..., p_n{}^{1}]$
...	...	...	...
$[id_{v1}{}^{m}, ..., id_{vi}{}^{m}]$	$[id_{c1}{}^{m}, ..., id_{cj}{}^{m}$	$[id_{o1}{}^{m}, ..., id_{ok}{}^{m}]$	$[p_1^m, ..., p_n{}^{m}]$

где p - множество состоящее из id товаров.
Здесь уже можно посчитать map@k - его значение колеблется от **0.01** до **0.015**.

map@k зависит от порядка, поэтому необходимо учитывать релевантность товара из списка рекомендаций. Предлагается изменить таблицу следующим образом (операция Explode)

session_id	order	prediction	target
$session_1$	$order_1$	p_1^1	y_1^1
$session_1$	$order_1$	$p_2^1[0]$	y_2^1
...	...	...	...
$session_1$	$order_1$	$p_n^1[0]$	y_n^1
...	...	...	...
$session_m$	$order_m$	$p_1^m[0]$	y_1^m
...	...	...	...
$session_m$	$order_m$	$p_n^m[0]$	y_n^m

$$y_i^j = \begin{cases} 1, p_i^j \in order^j \\ 0, p_i^j \notin order^j \end{cases} \tag{6}$$

Если предлагаемый id находится в заказе для данной сессии, то ставим 1, если нет - 0.

Таким образом учитываем релевантность отдельного товара из рекомендации.

Обычно для задач ранжирования используется метрика $NDCG$, которая показывает качество ранжирования - отклонение полученных оценок релевантности от действительных. Метрика NCDG для сессии q, набора рекомендаций m_q и меток l_q определяется следующим образом:

$$NDCG(q_i) = \frac{DCG(q_i)}{IDCG(q_i)} \tag{7}$$

$$DCG(q_i) = \sum_{k=1}^{m_q} \frac{l_{qk}}{log_2(k+1)} \tag{8}$$

$ICDG(q_i)$ - это значение для правильного ранжирования.
К сожалению, NCDG нельзя оптимизировать, т.к. она недифференцируема, поэтому вводится гладкая аппроксимация данной метрики **YetiRank**:

$$\mathbb{L} = -\sum_{i,j} \omega_{ij} log \frac{e^{x_i}}{e^{x_i} + e^{x_j}} \tag{9}$$

Здесь (i, j) - индексы всех пар рекомендаций для сессии, ω_{ij} - вес пары рекомендаций, а x_i, x_j - предсказанные значения релевантности рекомендаций i и j соответственно.

Таким образом в качестве модели на втором уровне используется градиентный бустинг над деревьями, где минимизируется YetiRank. В итоге модель учится предсказывать значение релевантности для товара. После этого необходимо собрать рекомендации обратно в вектор, но уже с учётом предсказанного значения релевантности.
Рассмотрим пример, где для сессии было получено 4 *первичных предсказания*:

session_id	order	prediction	relevance score
1	$[2, 3, 1, 4]$	4	0.1
1	$[2, 3, 1, 4]$	5	0.0
1	$[2, 3, 1, 4]$	1	0.3
1	$[2, 3, 1, 4]$	2	0.4

Собирая это в итоговое предсказания для сессии:

session_id	order	prediction
1	$[2, 3, 1, 4]$	$[2, 1, 4, 5]$

Кривые обучения и итоговые метрики

Благодаря переходу к задаче ранжирования удалось в разы улучшить целевую метрику mAP@k.

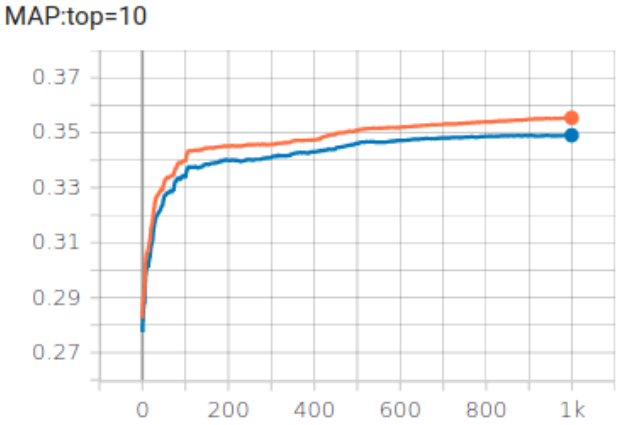


Рис.: map@10 (train/test)

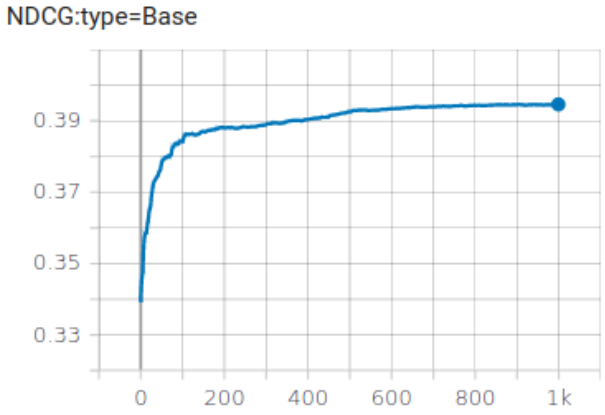


Рис.: NDCG

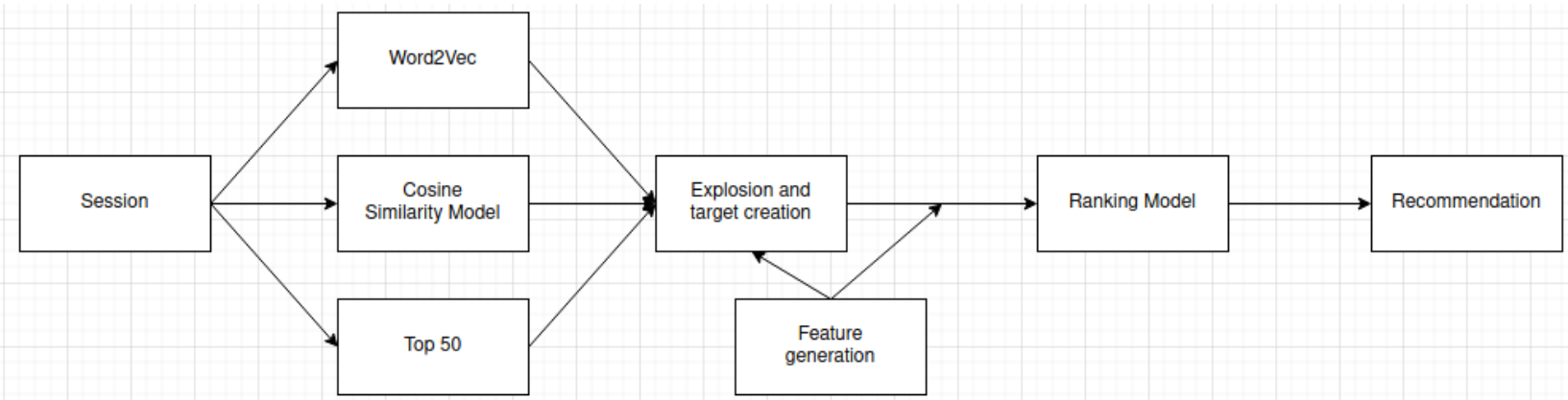


Рис.: Схема модели

[1] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin.
Catboost: gradient boosting with categorical features support.
arXiv preprint arXiv:1810.11363, 2018.

[2] Andrey Gulin, Igor Kuralenok, and Dimitry Pavlov.
Winning the transfer learning track of yahoo!'s learning to rank challenge with yetirank.
In *Proceedings of the Learning to Rank Challenge*, pages 63–76, 2011.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean.
Distributed representations of words and phrases and their compositionality.
Advances in neural information processing systems, 26:3111–3119, 2013.