МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра прикладной математики и
экономико-математических методов

Направление: Прикладная математика и информатика

ОТЧЕТ
по результатам решения кейса gpn-cup 2021
«Реализация рекомендательной системы»

Студента Скибин Максим Витальевич

Курс: 4 Группа: ПМ-1801
Форма обучения: Очная

Санкт-Петербург
2021 г.

СОДЕРЖАНИЕ

1. ПОСТАНОВКА ЗАДАЧИ

В качестве исходных данных представлены транзакционные данные продаж, содержащие следующие категории:

– уникальный идентификатор товара;

– цена, по которой был продан товар;

– количество товаров (если не топливо);

– количество литров (если товар – топливо);

– уникальный идентификатор чека;

– уникальный идентификатор клиента (если клиент «представился» при покупке);

– уникальный идентификатор магазина;

– дата транзакции.

И данные о товарах, содержащие следующие категории:

– уникальный идентификатор товара;

– полное наименование товара;

– наименование торговой марки;

– группа, к которой принадлежит товар;

– признак собственной торговой марки;

– единица измерения для количества;

– страна производства товара.

Задача состоит в реализации рекомендательной системы, которая будет предлагать покупателям 20 дополнительных товаров в чек из следующих групп:

– вода;

– сладкие газированные напитки, холодный чай;

– кофейные напитки с молоком;

– снеки;

– соки и сокосодержащие напитки.

В качестве метрики качества предложено использовать «Mean average precision at 20». Данная метрика строится на основе двух других метрик:

Метрика 1 - Precision at K

Количество релевантных элементов в рекомендации, если взять только верхние K рекомендованных товаров:

$$p@K = \frac{1}{K}\sum_{i=1}^{K} r_i^{true},$$

$$r_i^{true} = \begin{cases} 1, & \text{если на } i\text{-м месте стоит релевантнвый товар} \\ 0, & \text{в ином случае} \end{cases}$$

Метрика 2 – Average precision at 20

Средний $p@K$ для всех $K \in \{1, 2, \dots, 20\}$ в случаях, когда на K-м месте порекомендован релевантный товар:

$$ap@20 = \frac{1}{20}\sum_{K=1}^{20} p@K \cdot r_i^{true}$$

Метрика 3 – Mean average precision at 20

Средний $ap@20$ для всех N чеков тестовой выборки:

$$map@20 = \frac{1}{N}\sum_{i=1}^{N} ap@20_i$$

2. ПРЕДОБРАБОТКА ДАННЫХ

Все данные представлены в виде файлов расширения «parquet».

Транзакционные данные насчитывают 7 620 119 транзакций с пропусками в виде отсутствующих идентификаторов клиентов в половине транзакций. Данные представлены за 5 месяцев. В среднем по 1,5 миллиона наблюдений и пол миллиона уникальных покупок в месяц.

В данных о товарах содержится 5 103 уникальных товара с множеством пропусков в различных пунктах кроме уникального идентификатора товара и группы товаров.

Предобработка данных состоит из двух этапов:

1. Преобразование данных в множество файлов формата JSON.
2. Построение тестовой выборки для оценки качества моделей.

2.1. Преобразование данных в JSON файлы

Изначально предполагалось работать не с отдельными транзакциями, а с целыми чеками (множеством транзакций в одном чеке). Соответственно сама структура данных была преобразована и сохранена в группу JSON файлов. Для работы с JSON файлами было реализовано две функции:

1. Функция сохранения датасета в множество JSON файлов.
2. Функция чтения JSON файлов и преобразования данных в разреженную матрицу.

Функция сохранения датасета позволяет разбить один большой датасет на множество JSON файлов задаваемого количества. Ее программная реализация представлена на рисунке 1. Наличие данной функции позволяет не хранить в памяти весь большой датасет, а считывать его по частям при необходимости.

```python
def get_json(grouped_data, groups_names, foldername, n=20, k=None):
    # создадим отдельную директорию, если такой еще нет:
    if not 'json_data_'+foldername in os.listdir():
        os.mkdir('json_data_'+foldername)
    if k is None:
        k = n
    groups_amount = len(groups_names)
    batch_size = ceil(groups_amount / n)
    g_id = 0
    c = 0
    if not k is None:
        n = k
    # пройдем по каждому пакету:
    for bn in tqdm(range(k)):
        group_of_clients = {'group_id':g_id, 'clients':[]}
        # пройдем по каждому клиенту в пакете:
        for gn in tqdm(groups_names[bn*batch_size: bn*batch_size+batch_size]):
            group = grouped_data.get_group(gn)
            # соберем клиента:
            client = {'client_global_id': c,
                      'cheque_id': int(group['cheque_id'].values[0]),
                      'sku_ids': [sku for sku in group['sku_id']]}
            # ID клиента:
            try:
                client['client_id'] = int(group['client_id'].values[0])
            except ValueError:
                client['client_id'] = -1
            group_of_clients['clients'].append(client)
            c += 1
        # сохраним пакет в файл:
        file_name = 'json_data_'+foldername + '/' + 'group_' + str(g_id)
        with open(file_name+'.json', 'w') as write_file:
            json.dump(group_of_clients, write_file, indent=4)
        g_id += 1
    return 0
```

Рисунок 1 – Программная реализация функции сохранения датасета

Для построения первых моделей предполагалось работать с матрицами (Клиент-Товар). Элементы чтения и преобразования были объединены в одну функцию построения разреженной матрицы из данных. Программная реализация данной функции представлена на рисунке 2. Товары предварительно кодируются, каждому номеру товара ставится в соответствии номер столбца результирующей матрицы.

```python
def get_sparse_matrix(path, code_in):
    # отсортируем файлы с клиентами, чтобы не потерять порядок клиентов:
    json_names = sorted(os.listdir(path),
                        key=lambda x: int(x[6:].replace('.json', '')))
    goods_len = len(code_in.keys())
    indx_row_list, indx_col_list = [], []  # списки индексов по строке и столбцу
    values_list = []  # список значений
    # ID клиента и номер чека:
    clients_ids = []  # ID клиентов
    cheqs_ids = []  # ID чеков
    for js_name in tqdm(json_names):  # пройдем по каждому файлу
        file_path = path + '/' + js_name
        with open(file_path, 'r') as read_file:
            group_of_clients = json.load(read_file)
            for client in group_of_clients['clients']:  # по каждому клиенту
                # ID клиента и номер чека:
                clients_ids.append(client['client_id'])
                cheqs_ids.append(client['cheque_id'])
                # значения и индексы матрицы:
                indxs = defaultdict(int)
                for val in client['sku_ids']:  # сформируем покупки клиента
                    indxs[code_in[val]] += 1 / goods_len  # + нормировка
                for i in indxs:  # соберем индексы и значения для матрицы
                    indx_col_list.append(i)
                    values_list.append(indxs[i])
                indx_row_list.extend([client['client_global_id']]*len(indxs))
    # сформируем разреженную матрицу:
    sparse_matrix = sparse.coo_matrix(
        (np.array(values_list, dtype=np.float32), (indx_row_list, indx_col_list)),
        shape=(client['client_global_id']+1, goods_len))
    return sparse_matrix.tocsr(), clients_ids, cheqs_ids
```

Рисунок 2 – Программная реализация функции построения разреженной матрицы

2.2. Разбиение выборки на обучающую и тестовую

Для формирования тестовой выборки были взяты транзакции за последний месяц. Из списка транзакций был сформирован список клиентов путем закрепления за каждым чеком уникального клиента (даже если из данных известно, что некоторое количество чеков принадлежит одному клиенту). Для тысячи случайных клиентов из получившегося списка было найдено 25 «похожих» (в смысле покупок) клиентов. Количество похожих клиентов выбиралось небольшим с той целью, чтобы не «размазывать» индивидуальные

предпочтения одного клиента. По покупкам похожих клиентов определялся набор из двадцати наиболее часто покупаемых целевых товаров. В случае, когда количество найденных товаров меньше двадцати, получившийся набор дополнялся случайными целевыми товарами, не вошедшими в данный набор.

Естественно, что от построения тестовой выборки будут зависеть показатели метрик для моделей. К построению тестовой выборки можно подходить по-разному. Имеет место вариант, когда необходимо предсказать следующую покупку клиента. Но данный вариант был рассчитан недостаточным ввиду специфики задачи. Общее число целевых товаров – 974, 20 из которых необходимо порекомендовать. Не факт, что клиент покупал хотя бы какие-то 20 товаров из целевых. Но вполне разумно рекомендовать ему похожие товары, которые, например, покупают «похожие» клиенты, что и было взято за основу формирования тестовой выборки.

Количество уникальных чеков (клиентов) за целый месяц составляет пол миллиона. Таким образом размер разреженной матрицы (Клиент-Товар) составил $(575\,453 \times 5103)$. Для уменьшения размерности получившейся матрицы применялось сингулярное разложение. Ближайшие соседи искались уже в преобразованной матрице размера $(575\,453 \times 128)$. Получившаяся тестовая выборка была сохранена в отдельный JSON файл.

Для тренировочной выборки были взяты 200 тысяч уникальных клиентов соответственно из оставшихся четырех месяцев.

3. ПОСТРОЕНИЕ МОДЕЛЕЙ

3.1. Первые модели

В данном блоке представлены результаты применения отдельных моделей для решения поставленной задачи.

3.1.1. Baseline

В качестве базовой модели было принято предлагать каждому клиенту 20 наиболее популярных целевых товаров из всех транзакций. Данная модель является наименее персонализированной. Значение метрики для данной модели составило 0,000058. Данное значение является отправной точкой в улучшении качества рекомендательной системы.

3.1.2. ALS

Модель ALS (Alternating Least Squares) основана на разложении матрицы (Клиент-Товар). Для реализации применялась библиотека Implicit. Значение метрики для данной модели составило 0,000083, что является небольшим улучшением метрики относительно базовой модели – коэффициент прироста относительно базовой модели: 1,43 На самом деле уже на данном этапе было применено объединение двух моделей. Дело в том, что количество рекомендаций для каждого клиента было зафиксировано числом 974 (количество всех целевых товаров). Но может реализоваться случай, когда в данном рекомендованном наборе нет 20 целевых товаров. В таком случае недостающие товары заполнялись наиболее популярными целевыми товарами – то есть из базовой модели.

3.1.3. Item2Item

Данный метод основан на расчете «похожести» между купленными товарами. Для каждого товара составляется список тех товаров, которые наиболее часто покупают вместе с ним. Похожесть рассчитывается на основании косинусного расстояния. Данный метод показывается себя хорошо в случае, когда товаров существенно меньше, чем клиентов, что справедливо в данной задаче.

В данной модели параметром является количество похожих товаров, подбираемых для каждого товара в списке покупок клиента. Влияние данного параметра на значение метрики представлено на рисунке 3.

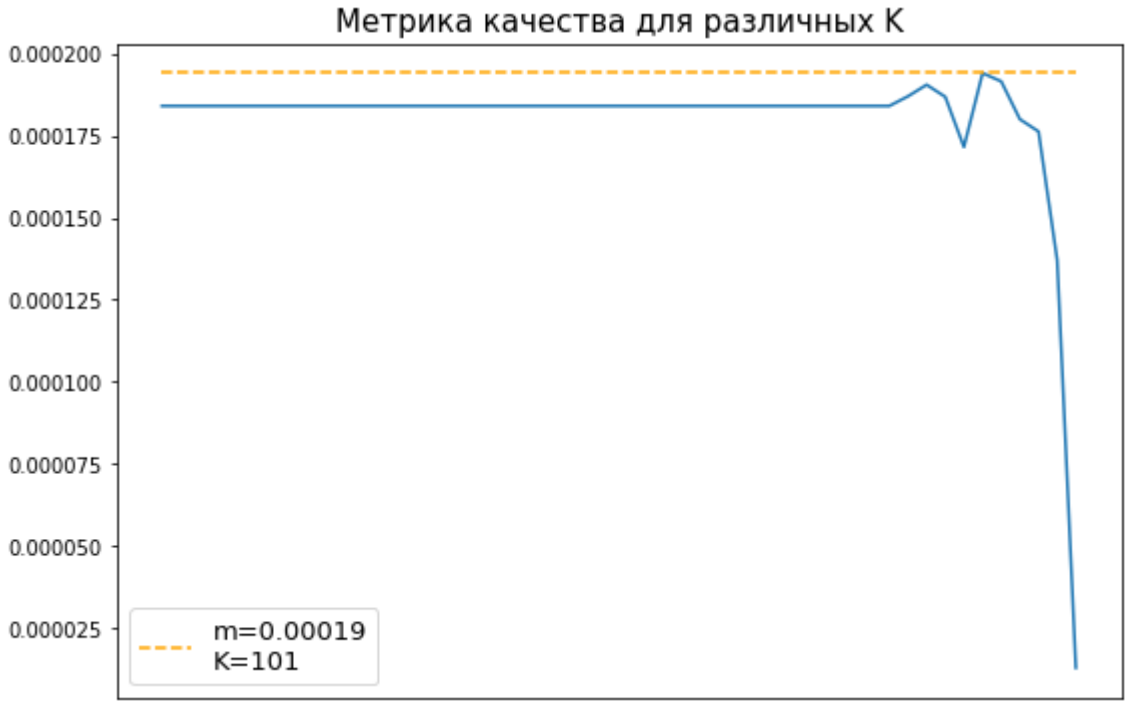

Рисунок 3 – Зависимость метрики качества от количества похожих товаров

Вдоль абсциссы количество соседей K варьируется от 981 до 1 с шагом 20. Из всех данных значений оптимальным является $K = 101$. Значение метрики в таком случае составляет 0.000194. Внутри модели аналогично предыдущей применяется также и базовая модель. Кратность улучшения значения метрики – 3,33 раза.

3.1.4. User2User

По сути своей модель User2User повторяет методологию построения тестовой выборки – поиск похожих клиентов и рекомендация товаров, которые они покупают. На практике же данная модель сработала хуже базовой модели. При количестве соседей, равном 256, значение метрики составило 0,0000808, а кратность прироста относительно Baseline – 1,39.

В предположении, что большое число соседей сильно «размазывает» предпочтения обособленного клиента, было реализовано варьирование

параметра числа соседей. Результаты по динамике метрики с уменьшением числа соседей представлены на рисунке 4.

Рисунок 4 – Зависимость метрики качества от числа похожих клиентов.

Из графика видно, что при меньшем количестве соседей метрика принимает высокое значение относительно других экспериментов в рамках данной модели. Значение метрики: 0,00012, кратность прироста: 1,98.

3.2. Комбинации моделей

В данном блоке представлены попытки сделать рекомендательную систему более персонализированной за счет объединения моделей. Для тех клиентов, которые «представлялись» системе при покупках, на основе тренировочной выборки составлена матрица (Клиент-Товар). К этой матрице применялись некоторые методы восстановления предпочтений по исходной матрице. На этапе прогнозирования (предложения товаров клиенту) полагалось поступать следующим образом:

1. Если клиент «представился» системе, и он имеет историю транзакций, то ему предлагаются продукты на основе его истории покупок. При необходимости персонализированные рекомендации дополняются рекомендациями из другой модели.

2. В случае же когда клиент не «представился» системе или данный клиент является новым клиентом (с точки зрения отсутствия истории транзакций), то в таком случае рекомендации строятся на не персонализированной модели.

Далее под словами «клиента нет в базе» будем иметь в виду либо буквальное отсутствие клиента в предыдущих транзакция, либо нового клиента, информации о покупках которого не было в принципе.

3.2.1. Модель Байеса и Item2Item (BI2I)

Рассмотрим архитектуру модели. В случае, когда клиента нет в базе – действия тривиальны – применяем модель Item2Item. Случай же, когда в базе присутствует история покупок клиента, рассмотрим подробнее.

Для каждого известного (с точки зрения наличия ID) клиента рассмотрим матрицу (Клиент-Товар), перебрав все товары, которые клиент когда-либо покупал. Для заполнения пропусков в предпочтениях воспользуемся моделью Байеса. Таким образом для каждого известного пользователя будет иметься список его предпочтений по каждому товару. С другой стороны, для этого же пользователя будем рекомендовать похожие товары с помощью модели Item2Item, которая также вернет список товаров и соответствующих им предпочтений. Заметим, что данные предпочтения никак нельзя друг с другом сравнивать. Скажем, $0{,}7$ по модели Байеса и $0{,}6$ по модели Item2Item в рамках двух разных товаров не говорит о том, что первый товар предпочтительнее так как $0{,}7 > 0{,}6$. Первый товар предпочтительнее только в рамках модели Байеса относительно товаров с меньшей степенью предпочтения. Таким образом возникает проблема – не понятно, как правильно комбинировать товары по предпочтениям, полученным из разных моделей.

Рассмотрим следующий вариант решения данной промежуточной задачи. Введем параметр α – уровень доверия к модели Байеса. Соответственно, значение $1 - \alpha$ – уровень доверия к модели Item2Item. Перенормируем получившиеся уровни предпочтений товаров из моделей с учетом данного

параметра α. Результаты нормировки будем сравнивать между собой для ранжирования товаров по предпочтительности.

В случае, когда модели предлагают одинаковые товары, будем нормировать их веса с учетом параметра α и в итоговый набор предпочтений брать полу сумму нормированных весов.

Динамика значения метрики при варьировании параметра α от 0,01 до 1 представлена на рисунке 5.

Рисунок 5 – Зависимость метрики качества от параметра α

Таким образом наибольшим значением метрики оказалось 0,000149 при параметре α равном 0,93.

ЗАКЛЮЧЕНИЕ

Результаты по применению отдельных моделей представлены в таблице 1 в виде значения метрики и прироста относительно базовой модели.

Таблица 1 – Сравнение качества работы моделей исходя из метрики

Модель	Метрика	Прирост
Baseline	0,000058	–
ALS	0,000083	1,43
Item2Item	0,000194	3,33
User2User	0,000115	1,98
BI2I	0,000149	2,56

Модель Item2Item показала себя наилучшим образом среди остальных моделей с точки зрения значения метрики. Для представления результатов были применены две модели – Item2Item и BI2I.

Возможные направления доработки рекомендательной системы включают в себя следующие аспекты.

1. Не была использована дополнительная информация о товарах и транзакциях. К примеру, в модели можно было бы включить время между покупками каждого известного клиента. Это могло бы уточнить поиск похожих клиентов.

2. Были рассмотрены исключительно модели, направленные на работу с матрицей (Клиент-Товар). Было бы хорошей мыслю внедрить другие подходы к построению рекомендательной системы и рассмотреть различные комбинации моделей.

3. В качестве моделей, которые можно было бы комбинировать, были рассмотрены лишь две модели, более того, ориентированные на один и тот же подход построения рекомендаций – на работу с матрицей (Клиент-Товар).

4. Был рассмотрен всего один вариант комбинации предложенных предпочтений разными моделями – на основе весов и нормировки. Возможно, ту

или иную модель стоило применять в зависимости от временного периода, когда были совершены транзакции клиента, от частоты покупок и прочего.