МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра прикладной математики

Допустить к защите
Заведующий кафедрой
д-р физ.-мат. наук, профессор
_____М.Х.Уртенов
_____2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

СТАТИСТИЧЕСКИЙ АНАЛИЗ И ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ УВЕЛИЧЕНИЯ ПРОДАЖ ДЛЯ ИНТЕРНЕТ-МАГАЗИНА

Работу выполнил _____ В.Л. Новиков

Направление подготовки 09.03.03 Прикладная информатика

Направленность (профиль)Прикладная информатика в экономике

Научный руководитель
канд. физ.-мат. наук, доцент_____ В.Н. Кармазин

Нормоконтролер
канд. физ.-мат. наук _____ Г.В. Калайдина

Краснодар
2018

РЕФЕРАТ

Работа состоит из 52 страниц, 3 глав, 11 рисунков, 1 таблицы и 16 источников.

Ключевые слова: РЕКОМЕНДАЦИОННАЯ СИСТЕМА, ОЦЕНКА, ФИЛЬТРАЦИЯ, ПРОГНОЗ.

Объектами исследования в данной работе являются методы построения рекомендательных систем, реализуемые на основе базы данных интернет магазина.

Предмет исследования – алгоритмы, с помощью которых создается рекомендательная система.

Данная работа посвящена изучению и реализации некоторых подходов построения рекомендационных системы как одну из технологий увеличения продаж для интернет-магазинов, с помощью средств языка программирования Python, а также Microsoft Excel.

Целью работы является исследование теоретических и практических методов и подходов к реализации и проектированию рекомендационных систем.

В процессе работы были реализованы два базовых метода и один комбинированный подход. Проведено сравнение результатов работы всех методов.

Практическая значимость данной работы состоит в том, что изученные технологии можно внедрить в работу интернет-магазинов, видеохостингов и информационных ресурсов для увеличения продаж и увеличения аудитории интернет ресурсов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные условия экономической конкуренции требуют от предпринимателей создания все новых и новых технологий завлечения покупателей и увеличения продаж. Одной из таких технология являются рекомендательные системы. Данные системы привнесли новые способы взаимодействия обычных веб-сайтов со своими пользователями. На замену предоставления статической информации, когда потенциальные покупатели ищут и, вероятно, приобретают товары, рекомендательные системы увеличивают степень интерактивности, а также расширяют предоставляемые пользователю возможности. Рекомендательные системы формируют рекомендации независимо для каждого конкретного пользователя на основе его прошлых покупок и поисков, а также на основе запросов прочих пользователей. Также такие рекомендации увеличивают чек клиентов, посредствам предложения смежных товаров или товаров которые как-то связаны с его прошлыми приобретениями. Существует множество принципов проектирования рекомендательных систем, некоторые из них будут разобраны в данной работе.

1 Постановка задачи

1.1 Цель и задачи проекта

Целью данной работы является реализация гибридного метода для построения рекомендательной системы (РС), которая сможет рекомендовать товары на сайтах интернет-магазинов пользователям. Для достижения поставленной цели в работе были выделены следующие задачи:

- рассказать про базовые методы построения рекомендационных систем и изучить их работу;
- построить рекомендательную систему с помощью метода ассоциативных транзитивных сетей;
- реализовать рекомендательную систему на основе метода коллаборативной фильтрации;
- провести сравнение реализованных методов;
- реализовать гибридную рекомендационную систему на основе нескольких методов и сравнить ее работу с базовыми подходами.

Объектами исследования в данной работе являются методы построения рекомендательных систем.

Предмет исследования – алгоритмы, с помощью которых создается рекомендательная система.

2 Теоретические аспекты рекомендательных систем

2.1 Основные понятия

Лента новостей и постов, которая находится в VKontakte или лента предлагаемых товаров в интернет магазинах уникальна для каждого пользователя. Почему же все остальные сайты одинаково выглядят для разных посетителей? Ответ в том, что социальные сети естественным образом персонализованы, а остальные сайты, обычно, нет.

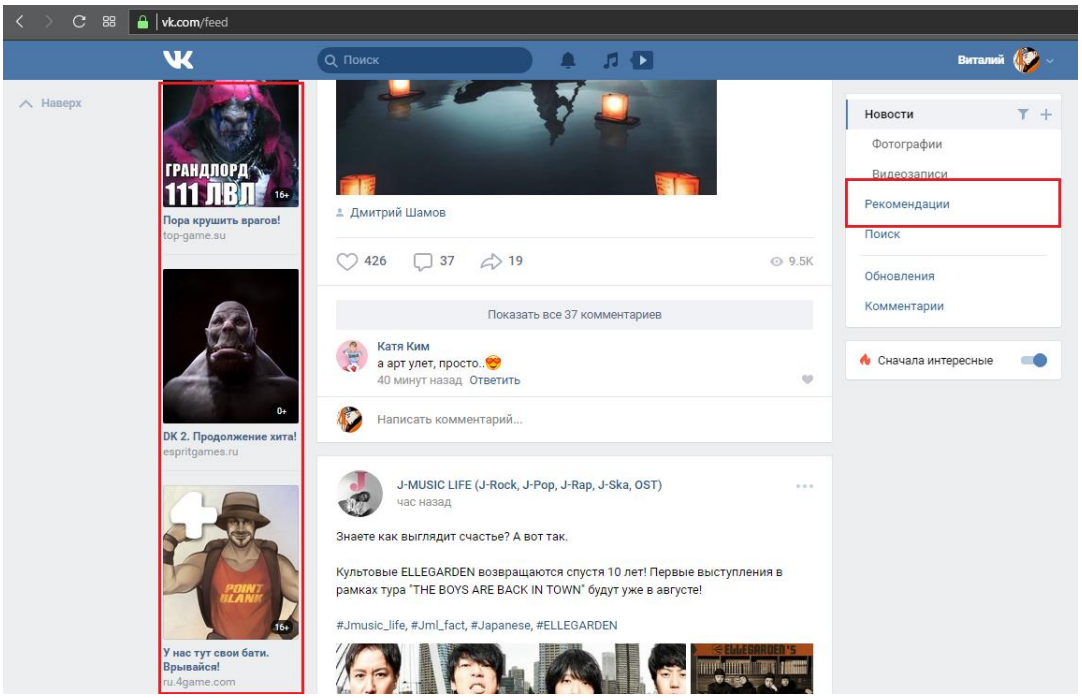

Рисунок 1 – Пример рекомендаций в социальной сети VK

Таким образом, персонализация интернет-сайта – это его автоматическая адаптация под необходимые текущие требования каждого клиента. В персонализации можно разделить два различных подхода.

Первый – это расширяющая персонализация, когда на основе некоторой информации о клиенте ему предлагается дополнительная информация, предположительно полезная для него. Ярким примером

являются товарные рекомендации, чаще всего встречающиеся в интернет-магазинах, например, с заголовком «С этим товаром также покупают».

Второй – это сужающая персонализация. Примером этой персонализации является алгоритм ленты новостей VKontakte, который из списков постов и записей друзей фильтрует и показывает пользователю только те записи, которые могут более всего привлечь его.

Эти подходы разрешают противоположные задачи.

Первый из них предлагает неизвестную ранее информацию. Второй – напротив, предотвращает перенасыщение, перегрузку информацией.

Рекомендательные системы — программы, которые пытаются предсказать, какие объекты (фильмы, музыка, книги, новости, веб-сайты) будут интересны пользователю, имея определенную информацию о его профиле. Существуют несколько подходов к разработке рекомендационных систем, которые применяются в зависимости от различных факторов:

- располагаемая информация о клиентах и рекомендуемых сущностях;
- типов явной и неявной ответной связи клиентов;
- предметной области [2].

Задача всех стандартных рекомендательных систем – это предложение пользователю товаров, о которых он ранее не знал, но которые могут оказаться полезными или интересными для него. Другими словами находится ответ на вопрос о том, какой продукт клиент пожелает купить в данное время. Имеется несколько вариантов рассмотрения этой задачи:

Гость – рассматривается вопрос, в какой степени он одобрил товары, которые попали в список рекомендаций. Показатели измерения данного вопроса это качество аппроксимации оценки и процент клиентов, которые купили товар из рекомендации.

Интернет-магазин – достижение максимальной прибыли с посетителя, а также наибольшая степень просмотра рекомендуемых товаров (чем дальше от списка к списку рекомендуемых товаров, тем лучше).

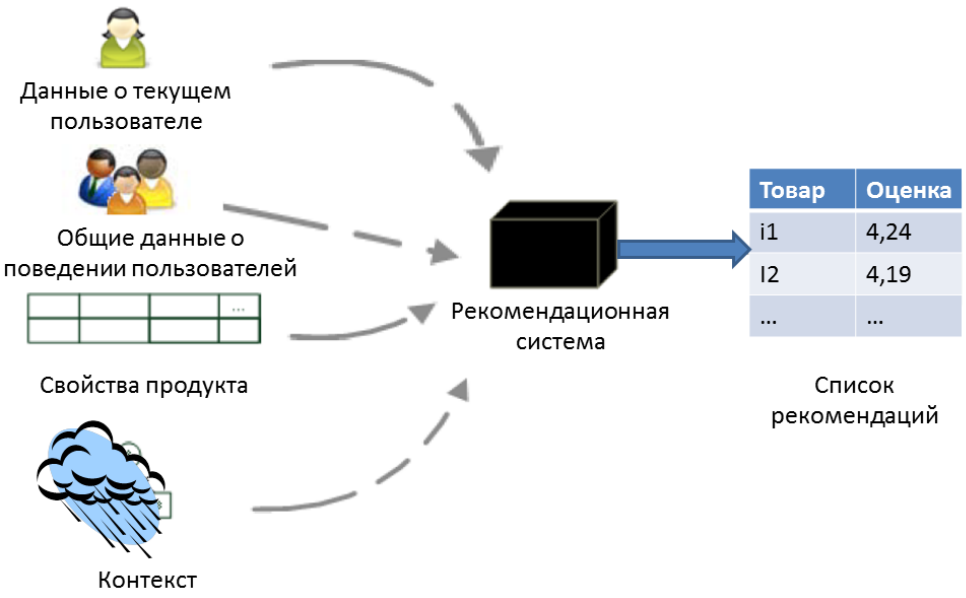

Рисунок 2 – Работа рекомендационной системы

Наиболее успешная РС чтобы сформировать список рекомендаций пользуется информацией не только об исследуемом клиенте, но и об остальных клиентах, а также о параметрах (различные характеристики: класс, форма, марка и т.д.) товаров, которые им рекомендуются (также учитывается сфера интересов клиента для рекомендации смежных товаров, комплектующих или товаров-заменителей). Для этой задачи нужно хранить и работать с очень массивными базами данных, эту проблему смогли решить совсем недавно и сейчас такие рекомендационные системы перестали быть редкостью [2].

2.2 Типы рекомендательных систем

2.2.1 Коллаборативная фильтрация

Этот метод формирует рекомендации, базируясь на характерных чертах ранее приобретенных товаров (модель поведения клиента, построенная на основе приобретенных им товаров ранее). Такая модель должна формироваться только на базе прошлого поведения клиента, но эффективнее всего учитывать также и истории запросов еще некоторого числа клиентов со схожими интересами. В тех случаях, когда коллаборативная фильтрация принимает во внимание поведение других пользователей, она использует информацию о группе (groupknowledge) для создания рекомендаций на основе похожести пользователей. По существу рекомендации основаны на автоматическом сотрудничестве множества клиентов и на выделении (с помощью фильтрации) тех пользователей, которые проявляют похожие предпочтения или шаблоны поведения.

В качестве примера предположим, что создается веб-сайт, с целью предложения его посетителям рекомендации относительно блогов или постов. Основываясь на данных от множества пользователей, которые подписываются на блоги и читают их, можно сгруппировать этих пользователей по их интересам. К примеру, можно объединить в одну группу пользователей, которые читают определенную группу блогов. По этой информации можно идентифицировать самые популярные блоги среди тех, которые читают участники этой группы. Потом для выбранного пользователя этой группы − рекомендовать самый популярный блог из тех, на которые он еще не подписан и которые он еще не читал.

В таблице на рисунке 3 строки соответствуют набору статей, а столбцы − посетителям. Ячейка на пересечении строки статьи и столбца посетителя содержит количество статей, которые были прочитаны этим посетителем в

этом наборе статей. Кластеризация пользователей за счет их привычек (например, с помощью алгоритма ближайшего соседа) позволяет выделить два кластера, содержащие по два пользователя. Стоит обратить внимание на похожесть литературных предпочтений у представителей каждого кластера: В кластер 1 входят участники Марк и Элиза, каждый из которых прочел по несколько статей по теме "Linux" и по теме "Облачные вычисления". В кластер 2 входят пользователи с именами Меган и Джилл, каждый из которых прочел по некоторое количество статей по теме "Java-технологии" и по теме "Agile-технологии".

Блоги	Марк	Меган	Элиза	Джилл
Linux	13	3	11	-
Продукты с открытым кодом	10	-	-	3
Облачные вычисления	6	1	9	-
Java-технологии	-	6	-	9
Agile-технологии	-	7	1	8
	Количество статей, прочитанных данным пользователем			
Кластер	1	2	1	2

Рисунок 3− Простой пример коллаборативной фильтрации

Сейчас можно идентифицировать определенные различия в границах текущего кластера и сформировать значимые рекомендации. В кластере 1 Марк прочел 10 постов из блога по продуктам с открытым исходным кодом, а Элиза не читала ни одной статьи по этой теме; Элиза прочла одну статью в блоге по Agile-технологиям, а Марк не читал ни одной статьи из этой темы. Поэтому, исходя из 3, Элизе можно порекомендовать блог по продуктам с открытым исходным кодом. А для Марка не представляется возможным сформировать никаких рекомендации, так как небольшая разница между ним и Элизой по количеству прочтений блога по Agile-технологиям с большой вероятностью будет отфильтрована. В кластере номер 2 Джилл прочитала три статьи в блоге по продуктам с открытым исходным кодом, а Меган не читала подобноых статей; Меган прочла 3 статьи в блоге по Linux, а Джилл

не рассматривала ни одной статьи по этой теме. Поэтому, для пользователей входящих в кластер 2 можно сформировать следующие рекомендации: пользователю Джилл порекомендуем блог по Linux, а пользователю Меган − блог со статьями по продуктам с открытым исходным кодом [2].

Следующий подход рассмотрения таких отношения основан на их сходствах и различиях, как показывает диаграмма Венна на рисунке 4. Сходства определяют, по каким признакам (с помощью нужного алгоритма) следует формировать группы пользователей, имеющих похожие предпочтнеия. Различия − это возможности, используемые для выработки рекомендаций – к примеру, с помощью применения фильтрации по популярности.

Рисунок 4 − Сходства и различия, используемые в коллаборативной фильтрации

На рисунке 4 показана упрощенная схема (которая страдает от разреженности данных по причине использования только двух примеров), такое представление весьма удобно.

2.2.2 Контентная фильтрация

Контентная фильтрация формирует рекомендацию на основе поведения посетителя. Например, данный подход может использовать ретроспективную информацию о просмотрах и запросах (какие статьи читает пользователь и некоторые характеристики таких статей). Если какой-либо пользователь обычно читает статьи о Linux или часто пишет комментарии в блогах по проектированию программного обеспечения, то контентная фильтрация может использовать данную ретроспективную информацию для обнаружения похожего контента и предложения этого контента в качестве рекомендации для нашего пользователя (статьи в блогах по Linux или в других блогах по проектированию программного обеспечения). Также такой контент может определяться в ручном режиме или выявлен автоматически с помощью других методов подобия.

Вернемся к рисунку 3 и рассмотрим посетителя по имени Элиза. Предположим, что применяется ранжирование блогов, которое указывает, что пользователи, прочитавшие статьи по Linux, могут быть заинтересованы облачными вычислениями и продуктами с открытым исходным кодом. В таком случае можно с легкостью порекомендовать Элизе − на основе ее текущих литературных предпочтений − блог по продуктам с открытым исходным кодом. Этот подход, показанный на рисунке 5, опирается только на контент, к которому обращается один пользователь, а не на предпочтения других пользователей в этой системе.

Рисунок 5 − Ранжированный список различий, используемый в контентной фильтрации

Диаграмму Венна можно применить и в этом случае: если одна сторона − это пользователь Элиза, а другая − это ранжированный набор подобных блогов, то сходства не учитываются (так как Элиза уже прочитала соответствующие блоги), а ранжированные различия предоставляют возможности для создания рекомендаций.

2.2.3 Гибридные подходы

Гибридные подходы, которые сочетают коллаборативную и контентную фильтрацию, могут повысить эффективность, точность (и сложность) рекомендательных систем. Простой пример гибридной системы мог бы использовать подходы, показанные на рисунке 3 и на рисунке 5. Если совместить результаты обоих фильтраций это может улучшить точность и качество рекомендаций. Также, гибридная РС становится очень актуальной, если применение коллаборативной фильтрации начинается при значительной разреженности данных (при холодном старте). Гибридный подход предоставляет возможность сначала взвешивать результаты согласно контентной фильтрации, а затем смещать эти веса по направлению к коллаборативной фильтрации (по мере "вызревания" доступного набора данных по конкретному пользователю)[2].

2.3 Описание основных алгоритмов рекомендательных систем

2.3.1 Корреляция Пирсона

Рассчитать меру похожести между изучаемым пользователем и всеми остальными (или их параметрами, например купленные товары или прочитанные статьи) можно при помощи корреляции Пирсона. Корреляция Писрона рассчитывает линейную зависимость между двумя объектами (пользователями) как функцию их параметров. Но этот алгоритм не вычисляет меру близости по всему набору объектов. Этот набор нужно заранее обработать, отфильтровав по убыванию показателя близости (высокоуровневый показатель схожести). Данный алгоритм очень часто используется при реализации коллаборативной фильтрации.

2.3.2 Алгоритмы кластеризации

Алгоритмы кластеризации − выявляют структуру в рядах случайных (без опознавательных меток) данных. Такие алгоритмы основаны на обнаружении сходства между объектами (например, между покупателями интернет магазина) посредством вычисления их расстояния от других элементов в пространстве признаков (feature space) (количество и качество приобретаемых товаров в одной транзакции). Количество независимых признаков определяет размерность пространства признаков. Если элементы сходны друг к другу, то они объединяются в кластер.

Самым простым из существующих алгоритмов кластеризации является алгоритм k-средних (k-means), который разделяет элементы на k кластеров. Первоначально элементы распределяются по этим кластерам в произвольном порядке. Затем для каждого кластера вычисляется центр масс (или просто центр) как функция его членов. Затем сравнивается расстояние каждого элемента кластера от центра этого кластера. Если элемент

оказывается ближе к другому кластеру, то он переходит в этот кластер. После проверки всех расстояний для всех членов центры кластеров вычисляются заново. Как только достигается устойчивое состояние (по результату проверки элементы не были перемещены) набор считается кластеризованным надлежащим образом, и затем алгоритм заканчивает свою работу.

Вычисление расстояния между двумя объектами может быть трудным для визуализации. Для решения этой проблемы следует рассматривать каждый элемент кластера как многомерный массив и вычислять для него т.н. евклидово расстояние.

Также существуют другие разновидности кластеризации, такие как: теория адаптивного резонанса (Adaptive Resonance Theory), нечеткая кластеризация методом C-средних (Fuzzy C-means), вероятностная кластеризация с помощью EM-алгоритма (Expectation-Maximization) и т. д.

Другие алгоритмы

В рекомендательных механизмах может применяться множество алгоритмов (а если учитывать вариации, то их еще больше). Наиболее применяемые алгоритмы:

- байесовские сети доверия (Bayesian Belief Nets)— визуально могут быть представлены как ориентированный ациклический граф, ребра которого представляют связанные вероятности переменных;
- цепи Маркова (Markov chains)— основаны на таком же подходе, как у байесовских сетей доверия, но решают проблему выработки рекомендации как последовательную оптимизацию, а не как простое прогнозирование;
- классификация по методу Роккио (Rocchio classification) (основанная на векторной модели) — использует отзывы о релевантности элементов для повышения точности рекомендаций.

2.3.3 Совместная фильтрация

В данном методе предполагается, что похожие пользователи делают схожие транзакции и похожие продукты приобретаются вместе.

Выделяют два основных подхода данного метода:

Фильтрация по пользователям (User-centric). В этом подходе неизвестная оценка товару ставится на базе оценок, которые были проставлены исследуемому товару клиентами, похожими на выбранного клиента. Этот подход реализуется двумя шагами:

1) найти пользователей, которые совершили аналогичные покупки, как и данный пользователь;

2) предложить товары с максимальным рейтингом среди товаров, выбираемых похожими пользователями.

Фильтрация по продуктам (Item-centric). Значение оценки продукта, которую необходимо определить, считается с помощью выборки схожих продуктов, включенных в одну запись пользователя. Данный метод включает в себя следующие шаги:

- построение матрицы продуктов для расчета степени схожести между ними;
- используя меру схожести предложить товары, которые похожи на заказанные ранее, данным пользователем.

Будем использовать следующие обозначения.

Предположим, что у нас есть информация о товарах (m) и пользователях (n). Транзакция i будет обозначаться m-мерным вектором оценок j-того продукта, который приобрели в данной транзакции, аналогично j-ый продукт обозначим n-мерным вектором оценок j-ого продукта во всех пользователях[5].

Фильтрация по пользователям (транзакциям).

$X_i, i \in \{1,...,n\}$ – множество нормально распределенных случайных величин, которые не зависят друг от друга. Лучшим (минимизация среднеквадратичной ошибки) прогнозом заказа j-ого продукта на u-ой транзакции, будет значение мат. ожидания соответствующей случайной величины, т.е. $EX_{u,j} = EX_{1,j}$, а оценкой этого прогноза – среднее арифметическое по выборке

$$\hat{x}_{u,j} = \frac{\sum_{i=1}^{n} x_{i,j}}{n}. \quad (1)$$

Число единиц заказанного j-ого товара входит в эту сумму с одинаковым весом для каждой i-ой транзакции, что соответствует нашему предположению о равноценности каждой транзакции. Метод фильтрации по пользователям заключается в том, что мы постулируем, что более похожие транзакции надо учитывать при прогнозировании с большим весом в сумме, чем менее похожие. При суммировании заказов j-ого продукта по всем транзакциям, должен учитываться вес $s(X,Y)$, который характеризует меру близости между транзакциями X и Y. Формулу (1) можно заменить (при прогнозировании размера заказа j-товара в u-ой (новой) транзакции) на формулу:

$$\hat{x}_{u,j} = \frac{\sum_{i=1}^{n} s_{trans}(X_i, X_u)\, x_{i,j}}{\sum_{i=1}^{n} \left| s_{trans}(X_i, X_u) \right|} \quad (2)$$

Чем больше сходство между транзакциями X_i и X_u, тем с большим весом входит число заказанного на i-ой транзакции j-ого товара во взвешенную сумму при прогнозе.

K ближайших соседа.

Метод состоит в нахождении ближайших транзакций (в количестве K) к данной транзакции. Затем использование среднего значения их оценок для предсказания неизвестных оценок. Мерой близости между транзакциями может служить, например, индуцированная евклидовая норма [5]. X_u – исследуемая транзакция. $X_i, i \in \{1,...,n\}$ – история уже выставленных рейтингов.

Будем обозначать матрицей $X \in Mat(n,m)$, где строки это транзакции, а столбцы Y_j, $j \in \{1,...,m\}$ рейтинги товары. $d(X_u, X_k) = \sqrt{\sum_{i=1}^{m} \left| \left(x_{u,j} - x_{k,j} \right)^2 \right|}$ – расстояние между транзакциями. Прогноз j-ого рейтинга в u-ой транзакции вычисляется как среднее значение j-ого рейтинга в K ближайших к x_u транзакциях

$$\hat{x}_{u,j} = \frac{\sum_{i \in I_K} x_{i,j}}{K},$$

где $I_K(X_u) \subset \{1,...,n\}$, состоит из K ближайших к x_u транзакций. Если использовать (2), то $\hat{x}_{u,j} = \frac{\sum_{i=1}^{n} s_{trans}(X_i, X_u) x_{i,j}}{\sum_{i=1}^{n} \left| s_{trans}(X_i, X_u) \right|}$, где функция близости

$$s_{trans}(X_u, X_i) := \begin{cases} 1, i \in I_K(X_u) \\ 0, i \notin I_K(X_u) \end{cases}. \quad (3)$$

Взвешенные ближайшие соседи

В данном методе для усреднения используется не только выбранное множество K, а все транзакции (взятые с весами, отражающими степень близости к исследуемой транзакции).

X_u – изучаемая транзакция, $X_i, i \in \{1,...,n\}$ – история совершенных транзакций. $d(X_u, X_k) = \sqrt{\sum_{i=1}^{m} (x_{u,j} - x_{k,j})^2}$ – расстояние между транзакциями. Веса, отражающие степень близости между транзакциями как величины обратные расстоянию – $s_{trans}(X_u, X_k) := \frac{1}{d(X_u, X_k) + \varepsilon}$,(4)

где $\varepsilon > 0$ малое число, которое служит, чтобы знаменатель не обращался в 0. Прогноз j-ого рейтинга в u-ой транзакции вычисляется как:

$$\hat{x}_{u,j} := \frac{\sum_{i=1}^{n} s_{trans}(X_u, X_i) x_{i,j}}{\sum_{i=1}^{n} |s_{trans}(X_u, X_i)|}$$

Метрики, основанные на углах между векторами

Для определения степени схожести между транзакциями используется корреляционный коэффициент Пирсона между m-мерными векторами транзакций (показатель линейной зависимости между центрированными векторами транзакций)

$$s_{trans}(X_u, X_i) := \frac{\sum_{k=1}^{m} (x_{u,k} - \bar{x}_u)(x_{i,k} - \bar{x}_i)}{\sqrt{\sum_{k=1}^{m} (x_{u,k} - \bar{x}_u)^2 \sum_{i=1}^{m} (x_{i,k} - \bar{x}_i)^2}}, \quad (5)$$

где $\bar{x}_i = \frac{\sum_{k=1}^{m} x_{i,k}}{m}$ среднее значение оценки по транзакции.

Также можно использовать косинус между m-мерными векторами транзакций (показатель линейной зависимости между векторами транзакций)

$$s_{trans}(X_u, X_i) = \frac{\sum_{k=1}^{m} x_{u,k} x_{i,k}}{\sqrt{\sum_{k=1}^{m} x_{u,k}^{\ 2} \sum_{k=1}^{m} x_{i,k}^{\ 2}}} \quad (6)$$

Минимизация средней абсолютной ошибки прогноза получаются при использовании «нормированного косинуса»:

$$s_{trans}(X_u, X_i) = \frac{\sum_{k=1}^{m} (x_{u,k} - \bar{y}_k)(x_{i,k} - \bar{y}_k)}{\sqrt{\sum_{k=1}^{m} (x_{u,k} - \bar{y}_k)^2 \sum_{k=1}^{m} (x_{i,k} - \bar{y}_k)^2}}, \quad (7)$$

где $\bar{y}_k := \frac{\sum_{p=1}^{n} x_{p,k}}{n}$ − средний рейтинг по k-ому столбцу. Значения рейтинга нормируются средним значением по строке, при использовании нормированного косинуса, значения рейтингов нормируются средними значениями по столбцам [5].

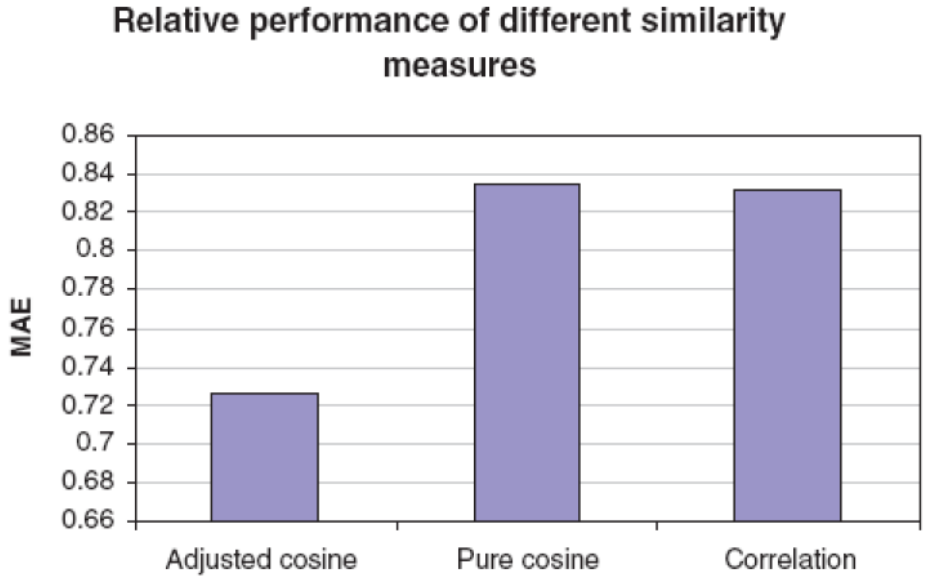

Рисунок 6 − относительная эффективность различных мер сходства

На практике можно понять, что лучшие результаты получаются в том случае, если при прогнозировании рейтинга вместо его значения использовать отклонение от среднего значения по транзакции, т.е.

$$\hat{x}_{u,j} := \bar{x}_u + \frac{\sum_{i=1}^{n} s_{trans}(X_i, X_u)(x_{i,j} - \bar{x}_i)}{\sum_{i=1}^{n} |s_{trans}(X_i, X_u)|} \quad (8)$$

С помощью (8) прогнозируется отклонение рейтинга j-ого товара от среднего рейтинга по транзакции.

Сложность и ресурсоемкость

Фильтрация по пользователям показывает довольно высокую степень точности в практических применениях. Однако, недостатком всех видов этого алгоритма является его требование к памяти и количество вычислений, требуемое для получения рекомендаций. А именно:

- если хранить в памяти (быстрый доступ) векторы рейтингов для всех транзакций, т.е. матрицу n строк на m столбцов, то для среднего интернет-магазина (примерно 1 млн. транзакций и примерно 10 тыс. товаров) потребуется хранить в памяти около 10 млрд. действительных чисел (по 8 байт), что не представляется возможным для имеющихся в распоряжении у таких магазинов компьютеров;
- осуществлять доступ к этим данным с диска (БД), осуществимо, но это сильно замедляет выполнение операций и, соответственно, увеличивает требования к аппаратному обеспечению (скорость доступа к дисковому вводу/выводу) и процессору (скорость выполнения арифметических операций);
- каждое сравнение выбранной транзакции с одной из n остальных занимает порядка O(m) операций, т.е. всего надо произвести $n \cdot O(m)$ операций для определения степени похожести между данной транзакцией и остальными;

- вычисление рейтинга для каждого из m товаров потребует выполнения порядка O(n) операций усреднения по транзакциям, т.е. $m \cdot O(n)$ операций для всех товаров;

- итого, нам потребуется, выполнить $m \cdot O(n) + n \cdot O(m) = O(mn)$ арифметических операций для получения рейтингов всех товаров для данной транзакции, чтобы получить рекомендации.

Выше приведенные рассуждения показывают, что фильтрация по пользователям может применяться только к относительно небольшим базам данных [10].

Фильтрация по товарам

Суть метода фильтрации по товарам состоит в выставлении неизвестного рейтинга товару в исследуемой транзакции на основании взвешенных рейтингов других товаров, входящих в эту транзакцию. Рейтинг товара получается тем больше, чем больше рейтингу других товаров в исследуемой транзакции, которые обычно покупаются с ним совместно. Т.е. если мы пытаемся проставить оценку товару А и нам уже известна оценка товара Б в этой транзакции, то оценка Б будет учитываться в вычислении оценки А с учетом того насколько часто А и Б входят в одну и ту же транзакцию. Мы получаем формулу, аналогичную (2):

$$\hat{x}_{u,j} := \frac{\sum_{i=1}^{m} s_{items}(Y_j, Y_i) x_{u,i}}{\sum_{i=1}^{m} \left| s_{items}(Y_j, Y_i) \right|} \tag{9}$$

Часто вместо абсолютной величины прогноза рейтинга товара прогнозируют отклонение рейтинга от средней по всем транзакциям для данного товара величину:

$$\hat{x}_{u,j} := \bar{y}_j + \frac{\sum_{i=1}^{m} s_{items}(Y_j, Y_i)(x_{u,i} - \bar{y}_i)}{\sum_{i=1}^{m} |s_{items}(Y_j, Y_i)|}, \quad (10)$$

где $\bar{y}_i := \frac{\sum_{k=1}^{n} x_{k,i}}{n}$ средний по всем транзакциям рейтинг i-ого товара.

Далее, аналогично фильтрации по пользователям, мы рассмотрим версии данного алгоритма в зависимости от способа вычисления функции близости между векторами рейтингов товаров Y_j и Y_i.

K ближайших соседа

Для j-ого n-мерного вектора рейтингов товара Y_j вычисляются K ближайших в евклидовой норме вектора $\{Y_{j1},...,Y_{jk}\}$ и определяется множество $I_K(Y_j) = \{j_1,..., j_k\}$, состоящее из индексов этих K ближайших к Y_j соседей. Тогда степень близости между векторами товаров Y_j и Y_i определяется как

$$s_{items}(Y_i, Y_j) = \begin{cases} 1, i \in I_K(Y_j) \\ 0, i \notin I_K(Y_j) \end{cases} \quad (11)$$

Т.е. усреднение в формуле (9) и (10) происходит только по K ближайшим:

$$\hat{x}_{u,j} = \frac{\sum_{i \in I_K(Y_j)} x_{u,i}}{K}$$

Взвешенные ближайшие соседи

Степень близости между векторами товаров в (9) и (10) определяется как величина, обратная евклидовому расстоянию между ними, т.е.

$$s_{items}\left(Y_i, Y_j\right) = \frac{1}{d\left(Y_i, Y_j\right) + \varepsilon} \tag{12}$$

Частота попарного вхождения

Весовые коэффициенты, учитывающие «близость» между n-мерными векторами рейтингов товаров вычисляются как относительные частоты совместного вхождения двух товаров в одну транзакцию:

$$s_{items}\left(Y_i, Y_j\right) = \frac{\sum_{k=1}^{n} \sigma\left(x_{k,i}, x_{k,j}\right)}{n}, \text{ где} \tag{13}$$

$\sigma\left(x_{k,i}, x_{k,j}\right) = \begin{cases} 1, x_{k,i}, x_{k,j} > 0 \\ 0, x_{k,i}, x_{k,j} = 0 \end{cases}$ – флаг того, что i-ый и j-ый товар совместно входят в k-ую транзакцию.

Метрики на основе углов между векторами товаров.

В качестве меры близости векторов товаров используется также корреляция Пирсона, косинус или нормированный косинус угла между ними:

$$s_{items}\left(Y_i, Y_j\right) = \frac{\sum_{k=1}^{n}\left(x_{k,i} - \bar{y}_i\right)\left(x_{k,j} - \bar{y}_j\right)}{\sqrt{\sum_{k=1}^{n}\left(x_{k,i} - \bar{y}_i\right)^2 \sum_{i=1}^{n}\left(x_{k,j} - \bar{y}_j\right)^2}},$$

$$s_{items}\left(Y_i, Y_j\right) = \frac{\sum_{k=1}^{n} x_{k,i}\, x_{k,j}}{\sqrt{\sum_{k=1}^{n} x_{k,i}^{\,2} \sum_{i=1}^{n} x_{k,j}^{\,2}}}, \tag{14}$$

$$s_{items}\left(Y_i, Y_j\right) = \frac{\sum_{k=1}^{n}\left(x_{k,i} - \bar{x}_k\right)\left(x_{k,j} - \bar{x}_k\right)}{\sqrt{\sum_{k=1}^{n}\left(x_{k,i} - \bar{x}_k\right)^2 \sum_{i=1}^{n}\left(x_{k,j} - \bar{x}_k\right)^2}},$$

где $\bar{y}_i = \frac{\sum_{k=1}^{n} x_{k,i}}{n}$ − среднее значение рейтинга по i-ому товару, а $\bar{x}_k = \frac{\sum_{p=1}^{m} x_{k,p}}{m}$ − среднее значение рейтинга по k-ой транзакции [10].

Сложность и ресурсоемкость

В отличие от фильтрации по транзакциям, где вычисление степени близости анализируемой транзакции ко всем остальным транзакциям может производиться только в реальном времени, так как данные о текущей транзакции становятся доступными только в момент формирования рекомендаций.

2.4 Проблемы рекомендательных систем

Возможности сбора информации, предоставляемые Интернетом, существенно облегчили использование "мудрости толпы" при помощи коллаборативной фильтрации. Но очень большое количество доступных данных усложняет реализацию данной возможности. Например, поведение некоторых пользователей хорошо поддается моделированию, но другие пользователи не демонстрируют устойчивого поведения. Большое количество таких пользователей может приводить к снижению точности результатов рекомендательной системы и к уменьшению ее эффективности. Кроме этого, пользователи могут задействовать рекомендательную систему для повышения значимости одного товара относительно другого товара — к примеру, посредством отправки положительных отзывов об одном товаре и отрицательных отзывов о его конкурентах. Хорошо спроектированная рекомендательная система обязана справляться с этими проблемами.

Существует еще одна проблема, присущая большим рекомендательным системам, связана она с масштабируемостью. Базовые алгоритмы хорошо работают со сравнительно небольшими объемами данных, но с увеличением этих наборов получение результатов на прежнем уровне качества при помощи базовых алгоритмов может стать проблемой. В случае оффлайновой обработки это может не составлять большой проблемы, но для сценариев реального времени необходимы более специализированные подходы.

Некоторые проблемы порождает необходимость соблюдения правил конфиденциальности. Рекомендательные алгоритмы способны распознавать различные закономерности, о существовании которых пользователи могут даже не подозревать. Пример такой ситуации имел место в крупной компании, которая смогла вычислять индекс прогнозирования беременности на основе покупательских предпочтений. После получения целевых рекламных объявлений отец дочери-подростка с удивлением узнал о ее беременности. Прогнозирующая система этой компании оказалась настолько точной, что смогла предсказать ожидаемую дату родов у будущей мамы на основе приобретаемых ею товаров [11].

3 Реализация выбранного проекта

3.1 Описание инструментов проектирования и реализации

Все рассматриваемые подходы создания рекомендационных систем будут реализованы на языке программирования Python версии 3.6. Данный язык программирования был выбран, потому что он располагает множеством как встроенных инструментов, так и скачиваемыми бесплатными библиотеками для работы с многомерными массивами. В данной работе были использованы такие библиотеки как: numpy, pandas, а также модуль math. Помимо большого количества инструментов, сам язык является простым для обучения и активно развивающимся.

Базы данных, которые используются для вычислений рекомендации, представлены в виде csv-файлов и импортируются в программу с помощью команды: pd.read_csv('d:/matrix.csv', ';', header=None)

G33 0

	A	B	C	D	E	F	G
1	товары/пользователи	Телефон Nokia	Наушники Philips	Чехол Хром	Зарядное устройство Sony	Брелки	Телевизор LG
2	Иванов(Исследуемый пользователь)	0	0	1	0	4	2
3	Сидоров	4	0	3	1	5	0
4	Новиков	1	4	2	1	4	5
5	Петров	2	4	2	1	0	4
6	Щукина	5	5	0	4	5	2
7	Васильев	1	0	1	0	2	2
8	Ляхов	4	5	4	4	3	3
9	Степанов	2	3	0	4	2	5
10	Ленина	5	1	4	4	2	4
11	Осипов	5	5	4	0	2	0
12	Кондратьев	1	5	5	1	0	2
13	Геращенко	2	5	2	1	4	0
14	Остапов	4	1	4	3	3	4
15	Алексеев	2	3	0	1	2	5
16	Михайлов	2	0	5	1	2	0
17	Левтеев	5	1	1	0	0	5
18	Понтелеев	0	0	3	4	4	3
19	Шизов	4	4	3	5	1	2
20	Акофьев	0	5	2	0	3	4
21	Минаев	1	4	0	5	1	0
22	Прокафьев	0	2	1	4	3	0
23	Андреев	5	2	1	2	3	5

Рисунок 7 – Фрагмент используемой выборки

3.2 Транзитивные ассоциативные сети

Идея этого подхода может быть проиллюстрирована на следующем примере. Предположим, что пользователи c1 и c2 приобрели товар p1, а пользователи c2 и c3 приобрели товар p2. Стандартные алгоритмы совместной фильтрации свяжут пользователей c1 и c2, а также пользователей c2 и c3, но не свяжут c1 и c3. Для получения транзитивных связей между транзакциями используется граф, узлы которого состоят из двух частей – транзакции товары, а дуги связывают транзакции с входящими в них товарами.

Соответствующая матрица связности имеет вид:

	p1	p2	p3	p4
c1	0	1	0	1
c2	0	1	1	1
c3	1	0	1	0

Положим, что наша задача заключается в формировании рекомендации товаров для пользователя в транзакции c1. Совместная фильтрация выработает рекомендации на основании похожести между транзакцией c1 с c2 и c3. Сходство c1 с c2 очевидно, потому что в обе транзакции входят товары p2 и p4. В результате для транзакции c1 будет рекомендован товар p3, потому что он был приобретен в транзакции c2. Сходство между транзакциями c1 и c3 традиционным алгоритмом совместной фильтрации не будет найдено, и потому товар p1, которой входит в c3, не будет рекомендоваться для c1.

Для учета транзитивных связей между транзакциями должны рассматриваться все пути между ними с длиной не превышающее заданного числа M. Обычные алгоритмы совместной фильтрации учитывают пути длиной три, например c1-p2-c2 или c1-p4-c2. Легко понять, что чем больше число различных путей, соединяющих два узла, тем значимее связь между

ними. Также интуитивно понятно, что чем длинней путь, связывающий два узла, тем слабее связь между ними. Мера близости между транзакцией с и товаром р определяется как сумма весов $\alpha(c,p)$ всех различных путей, соединяющих с и р. Вес каждого пути определяется как p^x, где $p \in (0,1)$ а $x \le M -$ длина пути. Таким образом, для нашего примера при $p = \frac{1}{2}$ мера близости между с1 и р3 будет равна $\alpha(c1,p3) = 2*\left(\frac{1}{2}\right)^3 = 0.25$ с учетом двух различных путей длины 3: с1-р2-с2-р3 и с1-р4-с2-р3. А мера близости между с1 и р1 будет равна 0,03125 для единственного пути длиной 5:с1-р2-с2-р3-с3-р1. Выходит, что рекомендательная сила товара р1 будет значительно меньше, чем р3, но в отличие от базовых алгоритмов совместной фильтрации, она не будет равна нулю.

Пусть дана матрица связности графа А, т.е. матрица, описывающая связи между продуктами и транзакциями. Тогда по индукции можно показать, что матрица достижимости за длину пути х (т.е. матрица А(х), элементами которой являются число различных путей длины х между продуктами и транзакциями) равна:

$$A(x) = \begin{cases} A, x = 1 \\ A \cdot A^t \cdot A(x-2), x = 3,5,7,... \end{cases} = \left(A \cdot A^t\right)^{\frac{x-1}{2}} \cdot A$$

Соответственно, матрица весов всех путей длины х, связывающих товары и транзакции будет равна

$$W(x) = p^x A(x) = p^x \left(A \cdot A^t\right)^{\frac{x-1}{2}} \cdot A$$

Так как матрица степеней сходства между товарами и транзакциями определена как сумма весов всех связывающих их путей с длиной, не превышающей М, то следует

$$\alpha(M) = \sum_{x=1}^{\frac{M+1}{2}} W(2x-1) = A \sum_{x=1}^{\frac{M+1}{2}} p^{2x-1} \left(A \cdot A^t\right)^{x-1}$$

Процесс выработки рекомендаций можно разделить на две фазы:

1. В отложенном режиме считаются матрицы, элементами которых являются количество различных путей длины x, связывающие товары с товарами: $A(x) = (A^t A)^{\frac{x}{2}}$, матрицы суммы весов путей длины x: $W(x) = p^x (A^t A)^{\frac{x}{2}}$, матрицы степеней сходства между товарами $\alpha(M-1) = \sum_{x=1}^{\frac{M-1}{2}} p^{2x} (A^t A)^x$ как сумма весов всех путей между ними длины меньше M.

2. В реальном времени степень сходства между новой исследуемой транзакцией и исследуемым товаром равна сумме весов всех путей от всех товаров, входящих в исследуемую транзакцию, к исследуемому товару с поправкой на коэффициент p (длина пути увеличилась на 1). Т.е. сумме степеней сходства между исследуемым товаром и товарами, входящими в исследуемую транзакцию, умноженную на α. Эта величина равна суммам элементов матрицы $p\alpha(M-1)$, соответствующих исследуемому товару и товарам, входящим в исследуемую транзакцию. Рекомендуются товары, степень сходства которых с исследуемой транзакцией максимальна [6].

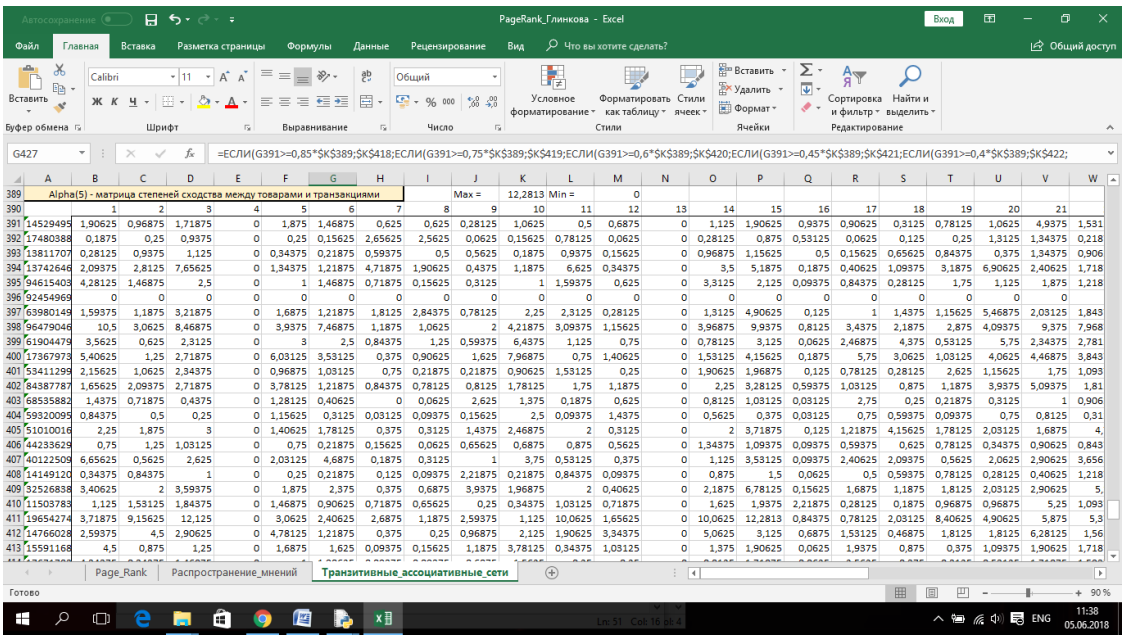

Рисунок 8 – Результаты работы транзитивных сетей(1)

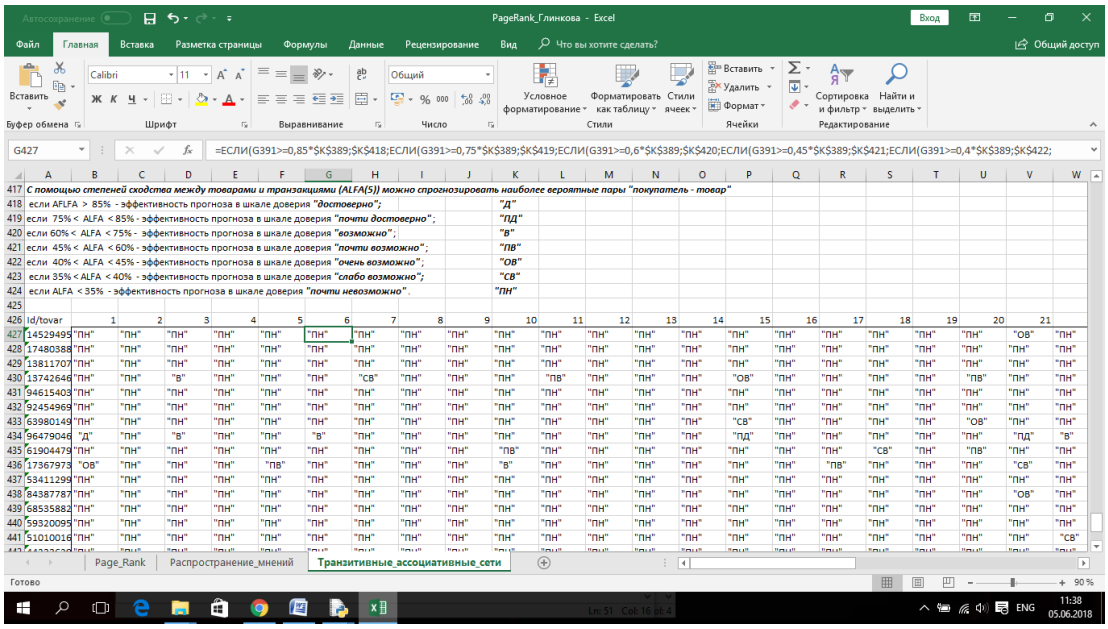

Рисунок 9 – Итоговая таблица прогнозирования пар товар/покупатель

Выбор критерия сравнения алгоритмов

Так как существует много различных параметров которыми можно оценить количественно точность работы рекомендационных систем, выбрать подходящий довольно сложно. Отсутствие определенных единых стандартов причиняет негативное влияние на развитие рекомендационных систем основанных на совместной фильтрации. Из-за этого продолжают вводить

новые показатели измерения для оценивания РС. При таком разнообразии используемых оценочных параметров становится сложно сравнивать результаты одного опубликованного исследования с результатами другого. В результате, становится тяжело интегрировать эти разные публикации в единое целое, чтобы выработать какие-либо общие знания и понятия относительно качества работы алгоритмов РС.

Точность

Тестирование на точность основано на случайном разделении транзакций на обучающее и тестовое множество (в пропорции 90%-10%, 80%-20%). Транзакции в обучающем множестве нужны для оценки рейтингов товаров из транзакций из тестового множества. Товары из каждой тестовой транзакции случайным образом разделяются на две группы: «известные» и «неизвестные». На основании рейтингов группы «известных» товаров строятся рейтинги для группы «неизвестных» товаров - на основании данных из множества обучающих транзакций. В качестве меры точности прогноза служит средняя абсолютная ошибка прогноза рейтингов MAE (MeanAbsoluteError):

$$MAE = \frac{\sum_{i \in I} \sum_{k \in K_i} \left| \bar{x}_{u,k} - x_{u,k} \right|}{\sum_{i \in I} \# K_i}$$ где – индексы тестового множества транзакций, а

K_i – множество «неизвестных» товаров в каждой тестовой транзакции [13].

3.3 Реализация коллаборативной фильтрации

Пусть имеется пользователь a. Наша задача – предсказать, какую оценку поставил бы пользователь a товару i. Будем рассматривать только пользователя a и пользователей, которые оценили товар i. Алгоритм состоит из трех шагов:

- для каждого пользователя вычисляем, насколько его интересы совпадают с интересами исследуемого пользователя;
- потом отбираем множество пользователей, наиболее близких к выбранному;
- предскажем оценку на основе оценок товара i похожими пользователями из предыдущего шага.

Первый шаг. Каждому пользователю в матрице R соответствует одна строка. Поэтому будем вычислять близость векторов-строк пользователей.
В данном случае используется формула нахождения косинуса между векторами (6).

Второй шаг. Выбираем множество K наиболее похожих на исследуемого пользователей. Можно попробовать выбрать всех пользователей, но так как пользователей с непохожими интересами может оказаться очень много, то они будут плохо влиять на точность предсказания оценки для товара i. Кроме того, выбранных количество пользователей влияет на объем вычислений на третьем шаге. Один из вариантов решений – установить порог меры близости, вычисленной на первом шаге. Пользователи с мерой близости, которые превышают заданный порог, войдут во множество K. Остальные – нет. Но обычно выбирается целая константа k. Затем все пользователи сортируются по убыванию меры близости. И во множество K входят k пользователей, наиболее близких к исслежуемому.

Третий шаг. Имея множество K близких пользователей нужно вычислить оценку, которую поставил бы пользователь a объекту i. Рассматриваются только те клиенты, которые оценили товар i. Нужная оценка вычисляется по формуле (9) [13].

```
============================== RESTART: D:\pi.py ==============================
Элементы с наиболее значимой мерой близости
0.8274010694405558
0.6907003710820745
0.6873690949183235
0.6631869522528651
0.6578208820621723
0.6548689819029808
0.6473929618545629
0.626541385090158
Номера отсортированных покупателей (по мере убывания похожести
5.0
10.0
32.0
19.0
4.0
7.0
29.0
36.0
S3=
[[5. 4. 1. 0. 4. 2. 3. 0. 5. 2. 1. 4. 3. 4. 0. 0. 2. 1. 2. 2. 0. 0.]
 [0. 5. 5. 5. 0. 2. 4. 5. 2. 2. 5. 5. 5. 4. 2. 0. 5. 0. 3. 3. 4. 0.]
 [5. 5. 2. 5. 5. 3. 4. 4. 5. 0. 0. 5. 3. 3. 3. 2. 0. 0. 5. 5. 2. 5.]
 [0. 4. 0. 5. 0. 0. 0. 2. 2. 5. 4. 5. 2. 4. 4. 2. 5. 4. 3. 4. 3. 0.]
 [5. 5. 0. 4. 5. 2. 5. 5. 5. 4. 2. 5. 0. 2. 0. 0. 4. 3. 0. 0. 3. 0.]
 [2. 3. 0. 4. 2. 5. 4. 4. 4. 4. 4. 5. 3. 0. 0. 0. 4. 3. 3. 0. 2. 0.]
 [5. 5. 5. 0. 4. 4. 0. 0. 0. 0. 0. 0. 2. 4. 3. 3. 5. 0. 5. 0. 3. 0.]
 [5. 4. 3. 5. 5. 0. 3. 0. 3. 5. 2. 3. 2. 5. 0. 3. 0. 4. 0. 0. 0. 2.]]
S4=
[0. 5. 2. 0. 4. 2. 0. 0. 4. 0. 0. 5. 4. 4. 0. 0. 3. 3. 2. 2. 2. 0.]
Среднее значение:
1.9090909090909092
Предсказанный результат для пользователя Иванов по товару Телефон Nokia:
2.7210855984067l9
pause...
```

Рисунок 10 – Результат работы коллаборативной фильтрации

В процессе работы алгоритма были найдены из списка пользователей те клиенты, которые больше всего похожи на исследуемого пользователя.

В качестве порога для выбора таких представителей была выбрана константа 8, так как если порог представить в виде процентного совпадения клиентов может оказаться очень много, что повлияет на сложность расчетов в третьем шаге.

Как видно из результатов работы коллаборативной фильтрации спрогнозированная оценка пользователя Иванов для товара “Телефон Nokia” равна примерно 2.7.

Таким образом, описанный алгоритм предсказывает оценки для объектов, которые текущий пользователь еще не оценил. Для того чтобы сделать рекомендацию для данного пользователя, достаточно предсказать

оценки для всех неоцененных объектов и выбрать объекты с наибольшей предсказанной оценкой [15].

3.4 Комбинированная фильтрация

Принцип комбинированной фильтрации заключается в получении оценки неизвестного рейтинга как взвешенной суммы оценок на основании фильтрации по транзакциям, фильтрации по товарам и смешенной фильтрации (на основании оценок похожих товаров в схожих транзакциях). Для получения рейтинга при смешенной фильтрации, т.е. на основании схожих товаров в похожих транзакциях, нам необходимо определить совместную степень сходства между парами транзакция - товар. Для этой цели используется величина, учитывающая степень сходства отдельно соответствующих транзакций и отдельно - товаров. Комбинированная степень сходства не превышает отдельных степеней сходства между транзакциями и товарами.

$$s_{trans-items} = \frac{1}{\sqrt{\left(\frac{1}{s_{trans}(X_u, X_i)}\right)^2 + \left(\frac{1}{s_{items}(Y_k, Y_j)}\right)^2}},$$

где S_{trans} вычисляется согласно (5), (6) или (7), а S_{items} – согласно (14), (15) или (16).

Под множеством $S_{trans}(X) \subset \{1,...,n\}$ будем обозначать индексы K самых близких с точки зрения (5), (6) или (7) к X транзакций из $X_1,...,X_n$. Под $S_{items}(Y) \subset \{1,...,m\}$ будем обозначать индексы K самых близких с точки зрения (14), (15) или (16) к Y товара из $Y_1,...,Y_m$. Сортируя и переиндексируя строки и колонки матрицы рейтингов по степени схожести к исследуемой транзакции и к исследуемому товару и оставив в ней для

последующего анализа не более K строк и столбцов, мы получим матрицу похожих транзакций и товаров (таблица 1) $M_{u,k} = \left\{ x_{i,j} \middle| i \in S_{trans}(X_u), j \in S_{items}(Y_k) \right\}$.

Степень сходства по товарам

Неизвестный рейтинг	Фильтрация по товарам	Фильтрация по товарам	Фильтрация по товарам	Фильтрация по товарам
Фильтрация по транзакциям	Смешанная фильтрация	Смешанная фильтрация	Смешанная фильтрация	Смешанная фильтрация
Фильтрация по транзакциям	Смешанная фильтрация	Смешанная фильтрация	Смешанная фильтрация	Смешанная фильтрация
Фильтрация по транзакциям	Смешанная фильтрация	Смешанная фильтрация	Смешанная фильтрация	Смешанная фильтрация
Фильтрация по транзакциям	Смешанная фильтрация	Смешанная фильтрация	Смешанная фильтрация	Смешанная фильтрация

Степень сходства по транзакциям

Таблица 1 – Структура матрицы после преобразований

Если для оценки неизвестного рейтинга мы будем учитывать только самый первый столбец с использованием меры близости по транзакциям, то получим оценку по транзакциям:

$$\hat{x}_{u,k}^{\ trans} = \frac{\sum_{i=1, j\neq u}^{K} s_{trans}(X_u, X_i) x_{i,k}}{\sum_{i=1, j\neq u}^{K} s_{trans}(X_u, X_i)} \qquad (18)$$

Если для оценки неизвестного рейтинга мы будем учитывать только первую строку с использованием меры близости по товарам, то получим оценку по товарам:

$$\hat{x}_{u,k}^{\ items} = \frac{\sum_{j=1, j\neq k}^{K} s_{items}(Y_j, Y_k) x_{u,j}}{\sum_{j=1, j\neq k}^{K} s_{items}(Y_j, Y_k)} \qquad (19)$$

Если для оценки рейтинга учитывать все элементы матрицы кроме первой строки и первого столбца, т. е. похожие товары, участвующие в похожих транзакциях, то используем формулу:

$$\hat{x}_{u,k}^{\ trans-items} = \frac{\sum_{i=1,i\neq u}^{K}\sum_{j=1,j\neq k}^{K} s_{trans-items}\left(x_{u,k}, x_{i,j}\right)x_{i,j}}{\sum_{i=1,i\neq u}^{K}\sum_{j=1,j\neq k}^{K} s_{trans-items}\left(x_{u,k}, x_{i,j}\right)} \qquad (20)$$

Итоговая комбинированная оценка неизвестного рейтинга получается в виде взвешенной оценки из (18), (19) и (20):

$$\hat{x}_{u,k} = \lambda(1-\delta)\hat{x}_{u,k}^{\ trans} + (1-\lambda)(1-\delta)\hat{x}_{u,k}^{\ items} + \delta\hat{x}_{u,k}^{\ trans-items}, \qquad (21)$$

где $\lambda, \delta \in [0,1]$ – веса учета соответствующих оценок. При $\delta = 0, \lambda = 1$ получим фильтрацию по транзакциям, при $\delta = 0, \lambda = 0$ – фильтрация по товарам, при $\delta \neq 0$ учитываем рейтинги сходных товаров и сходных транзакций. Результат предсказывания рейтинга для исследуемого пользователя и выбранного товара показан на рисунке 11 [14].

```
========================== RESTART: D:\Combin.py ==========================
Мера близости по убыванию
[0.9231861823449954, 0.8883363181483769, 0.8718127501157389, 0.8356290217967335,
 0.8206750043022921, 0.8104432008587534, 0.7861238160783628, 0.7756717518813397,
 0.7735659346940 95, 0.7717436331412898, 0.5773502691896258, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Множество К похожих пользователей и товаров
[[5. 4. 2. 3. 2. 2. 4. 4.]
 [5. 4. 2. 3. 2. 2. 4. 4.]
 [4. 5. 0. 3. 2. 5. 5. 5.]
 [5. 2. 5. 4. 3. 2. 4. 3.]
 [5. 4. 2. 3. 2. 2. 4. 4.]
 [5. 4. 2. 3. 2. 2. 4. 4.]
 [5. 4. 2. 3. 2. 2. 4. 4.]
 [5. 4. 2. 3. 2. 2. 4. 4.]]
Оценка по транзакциям
0.5017931349718318
Оценка по товарам
0.5052023498825421
Смешанная оценка
3.3324273054019313
Итоговая оценка товара Телефон Nokia для покупателя Иванова
3.1908955968804262
>>>
```

Рисунок 11 – Результаты работы программы

Результатом работы данной программы является предсказанный рейтинг товара, который пользователь может поставить если купит его.
В данном случае он составляет 3.2. Что является более точной оценкой по сравнению с коллаборативной фильтрацией.

3.5 Преимущества и недостатки

Плюсы реализованного гибридного метода:

- точность предсказания неизвестного рейтинга для товара
- возможность рекомендаций в более узких категориях продукции
- меньшая нагрузка на аппаратурные ресурсы, так как не нужно хранить много объемных массивов (благодаря заранее выбранному числу похожих массивов)

Минусы:

- не решена проблема “холодного старта”
- трудоемкость вычисления степени похожести и усреднения рейтингов по товарам или транзакциям.

ЗАКЛЮЧЕНИЕ

Перед каждым предприятием часто возникают вопросы о том, как увеличить чек покупателя или привлечь большую аудиторию к своему контенту или продукции. В этой работе мы изучили и реализованы рекомендационные системы как технология увеличения продаж для интрнет-магазина.

В ходе данной работы были следующие задачи:

- изучены базовые методы и особенности создания рекомендационных систем
- были изучены модули и библиотеки Python для работы с большими массивами данных (numpy и pandas)
- реализованы несколько видов рекомендационных систем на языке программирования Python
- проведено улучшение базового подхода путем совмещения нескольких подходов
- произведено сравнение базовых методов и комбинированного подхода
- выявлены преимущества и недостатки гибридного метода

Данный реализованный гибридный метод не был испытан на реальных интернет ресурсах, но исследования были проведены на основе данных приближенных к реальным условиям. Результат проведенной работы можно оценить как удовлетворительный. Были получены базовые знания и навыки для проектирования различных вариаций и комбинаций простейших методов создания рекомендационных систем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Системы рекомендаций: обзор современных подходов А.Г. Гомзин, А.В. Коршунов с 403 URL: https://cyberleninka.ru/article/n/sistemy-rekomendatsiy-obzor-sovremennyh-podhodov (дата обращения: 03.04.2018)

2 Рекомендательные системы. Часть 1. Введение в подходы и алгоритмы URL: https://www.ibm.com/developerworks/ru/library/os-recommender1/index.html (дата обращения: 01.04.2018)

3 Рекомендательные системы Часть 2. Механизмы с открытым исходным кодом URL: https://www.ibm.com/developerworks/ru/library/os-recommender2/index.html (дата обращения: 03.04.2018)

4 Язык программирования Python 3 для начинающих и чайников URL: https://pythonworld.ru/ (дата обращения: 04.04.2018)

5 Рекомендательные системы URL: http://businessdataanalytics.ru/download/RecommendationSystems.pdf (дата обращения: 05.05.2018)

6 Стандартная библиотека Python URL: https://ru.wikipedia.org/wiki Стандартная_библиотека_Python#CSV (дата обращения: 05.05.2018)

7 Библиотека numpy URL: http://www.numpy.org/ (дата обращения: 05.05.2018)

8 DATA MINING: РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ Collaboration Filtering Максим Гончаров URL: http://kek.ksu.ru/eos/WM/50_132-670.pdf (дата обращения: 13.04.2018)

9 NumPy Manual v.1.14 URL: https://docs.scipy.org/doc/numpy/index.html (дата обращения: 05.04.2018)

10 Алгоритмы кластеризации URL: https://https://basegroup.ru/community/articles/datamining (дата обращения: 06.06.2018)

11 Введение в Pandas: анализ данных на Python URL: https://khashtamov.com/ru/pandas-introduction/ (дата обращения: 03.04.2018)

12 Разрабатываем рекомендательную систему на Python [GeekBrains]! URL: https://www.youtube.com/watch?v=WV1ruv8aURg (дата обращения: 03.04.2018)

13 *Y. Koren, R. Bell, C. Volinsky* Matrix Factorization Techniques for Recommender Systems (дата обращения: 03.04.2018)

14 *Melville P., Mooney R., Nagarajan R.* Content-Boosted Collaborative Filtering for Improved Recommendations URL: http://www.aaai.org/Papers/AAAI/2002/AAAI02-029.pdf (дата обращения: 03.04.2018)

15 Netflix Recommendations: Beyond the 5 stars (Part 1) URL: https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429 (дата обращения: 03.04.2018)

16 Алгоритмы кластеризации на службе Data Mining URL: https://basegroup.ru/community/articles/datamining (дата обращения: 03.04.2018)

ПРИЛОЖЕНИЕ А

Код транзитивной ассоциативной сети

```
import pandas as pd
import numpy as np
from math import *
m = pd.read_csv('d:/matrix2.csv', ';', header=None)
mt = m.transpose()
#Находим матрицу A(x):
n = 3
Ax = np.zeros((40,40))
Ax1 = np.zeros((40,19))
AXmnoj = []
WXmnoj = []
P = 0.5
#Строки,столбцы,количество матриц
for x in range(3,8,2):
    Ax = (np.matmul(m,mt))**((x-1)/2)
    Ax1 = np.matmul(Ax,m)
    AXmnoj.append(Ax1)
    Wx = Ax1*P**x
    WXmnoj.append(Wx)
aM = np.zeros((40,19))
for g in range(3):
    aM += WXmnoj[g]
print ('Исходная матрица')
print (m)
print ('Транспонированная матрица')
print (mt)
```

```
    print ('Умножение исходной матрицы на
транспонированную и возведение в степень')
    print (Ax)
    print ('Умножение на исходную')
    print (Ax1)
    print ('Список получившихся матриц Ax')
    for i in range(3):
        print (AXmnoj[i])
    print ('Список получившихся матриц Wx')
    for j in range(3):
        print (WXmnoj[j])
    print ('матрица суммы всех путей')
    print (aM)
```

ПРИЛОЖЕНИЕ Б

Код коллаборативной фильтрации

```
import pandas as pd
import numpy as np
from math import *
#Открытие csv файлов
m1 = pd.read_csv('d:/matrix3.csv', ';', header=None)
#Создание массива для оценки близости клиентов
S = np.zeros((44))
S1 = np.zeros((44))
a = 0
#Шаг 1 Рассчет близости интересов
for u in range(1,44):
    d1 = 0
    d2 = 0
    d3 = 0
    for i in range(22):
        d1 = d1 + m1[i][a] * m1[i][u]
        d2 = d2 + m1[i][a]**2
        d3 = d3 + m1[i][u]**2
    if (d2 == 0 or d3 == 0):
        S[u] = 0
    else:
        S[u] = d1/(sqrt(d2) * sqrt(d3))
#print (S)
S1 = sorted(S, reverse = True)#Отсортированные SIM(a,u)
```

```
    print('Элементы с наиболее значимой мерой
близости')
    for g2 in range(8):
        print (S1[g2])
    #Вытаскиваем номера отсортированных мер близости
    S2 = np.zeros((44))
    for k in range(44):
        for j in range(8):
            if S1[j] == S[k]:
                S2[j] = k
    print('Номера отсортированных покупателей (по мере
убывания похожести')
    for g3 in range(8):
        print (S2[g3])
    # Количество взятых покупателей = 3 (n-1=2)
    S3 = np.zeros((8,22))
    S4 = np.zeros((22))
    for g in range(8):
        p = S2[g]
        for h in range(22):
            S3[g][h] = m1[h][p]
            S4[h] = m1[h][a]
    print ('S3= ')
    print (S3)
    print ('S4= ')
    print (S4)
    SrA = np.mean(S4)
    # Шаг 3 Прогноз оценивания пользователем продукта i
    i = 0
    otk = 0
```

```
    otk1 = 0
    for b in range(8):
        otk += (S3[b][i] - S3[b].mean())* S1[b]
        otk1 += abs(S1[b])

    Preds = SrA + otk / otk1
    print ('Среднее значение: ')
    print (SrA)
    print  ('Предсказанный результат для пользователя
Иванов по товару Телефон Nokia: ')
    print (Preds)
    input('pause...')
```

ПРИЛОЖЕНИЕ В

Код комбинированного метода

```
import pandas as pd
import numpy as np
from math import *
#Открытие csv файлов
m1 = pd.read_csv('d:/matrix3.csv', ';', header=None)
S = np.zeros((44))
S1 = np.zeros((22))
m1_sort = np.zeros((44,22))
U = 0
#Шаг 1 Рассчет близости интересов Strans
#i - Строки(транзакции) k - Столбцы(товары) U - исследуемая транзакция
for i in range(1,44):
    d1 = 0
    d2 = 0
    d3 = 0
    for k in range(22):
        d1 = d1 + m1[k][U] * m1[k][i]
        d2 = d2 + m1[k][U]**2
        d3 = d3 + m1[k][i]**2
    if (d2 == 0 or d3 == 0):
        S[i] = 0
    else:
        S[i] = d1/(sqrt(d2 * d3))
# Сортируем меру близости по убыванию(пользователи)
S12 = sorted(S, reverse = True)
```

```
    #print (S)
    #print('Отсортированный список по пользователям')
    #print (S12)
    # номера строк
    nom_str = np.zeros((8))
    for nom in range(8):
        for nom1 in range(8):
            if S12[nom1] == S[nom]:
                nom_str[nom1] = nom
    #Введем новую матрицу для сортировке по строкам
    Po_str = np.zeros((8,22))
    for nomer_stlb in range(22):
        for nomer_str in range(8):
            Poz = nom_str[nomer_str]
            Po_str[nomer_str][nomer_stlb]              =
m1[nomer_stlb][Poz]
    #  Sitems  Ui  -  исслед  товар  k1  -  товары  i1  -
транзакции
    U1 = 0
    for k1 in range(1,22):
        d11 = 0
        d21 = 0
        d31 = 0
       # print('Следующая итерация')
        for i1 in range(8):
            d11 = d11 + m1[k1][U1] * m1[k1][i1]
            d21 = d21 + m1[k1][U1] * m1[k1][U1]
            d31 = d31 + m1[k1][i1] * m1[k1][i1]
        if (d21 == 0 or d31 == 0):
            S1[i1] = 0
```

```
        else:
            S1[k1] = d11/(sqrt(d21 * d31))
    # Сотритруем меру близости по убыванию(товары)
    S11 = sorted(S1, reverse = True)
    print('Мера близости по убыванию')
    print (S11)
    # Номера столбцов
    nom_stolb = np.zeros((22))
    for nom2 in range(22):
        for nom3 in range(22):
            if S11[nom3] == S1[nom2]:
                nom_stolb[nom3] = nom2
    #print(nom_stolb)
    Po_stlb1 = np.zeros((8,8))
    for nomer_str1 in range(8):
        for nomer_stlb1 in range(8):
            Poz1 = int(nom_stolb[nomer_stlb1])
            #print(Poz1)
            Po_stlb1[nomer_str1][nomer_stlb1]          =
Po_str[nomer_str1][Poz1]
    print('Множество   К   похожих   пользователей   и
товаров')
    print(Po_stlb1)#   Матрица   отсортированная   по
убыванию столбцов и строк
    #print (m1)
    S2 = np.zeros((8,22))
    for g in range(8):
        for j in range(22):
            if (S12[g] == 0 or S11[j] == 0):
                S2[g][j] = 0
```

```
            else:
                S2[g][j]                                    =
1/sqrt((1/S12[g])**2+(1/S11[j])**2)
    #Оценка по транзакциям xtrans
    x_trans = 0
    #Координата поиска (0,0)   x(0,0)
    for l in range(8):
        sum1 = 0
        sum2 = 0
        for h in range(8):
            sum1 += S12[h]*S2[h][l]
            sum2 += S12[h]
            x_trans = sum1 / sum2
            break
    print('Оценка по транзакциям')
    print(x_trans)
    #Оценка по товарам xitems
    x_items = 0
    for h1 in range(8):
        sum3 = 0
        sum4 = 0
        for l1 in range(8):
            sum3 += S11[l1] * S2[h1][l1]
            sum4 += S11[l1]
            x_items = sum3 / sum4
            break
    print('Оценка по товарам')
    print(x_items)
    #Оценка по товарам/транзакциям xitems/trans
    sum5 = 0
```

```
    sum6 = 0
    for i3 in range(8):
        for j3 in range(8):
            sum5 += S2[i3][j3] * Po_stlb1[i3][j3]
            sum6 += S2[i3][j3]
    x_trans_items = sum5 / sum6
    print('Смешанная оценка')
    print(x_trans_items)
    # Итоговая комбинированная оценка
    alpha = 1
    betta = 0.95
    x_itog = alpha * (1 - betta) * x_trans + (1 - alpha) * (1 - betta) * x_items + betta * x_trans_items
    print('Итоговая оценка товара Телефон Nokia для покупателя Иванова')
    print(x_itog)
```