МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Институт Информационных Технологий
Кафедра Вычислительной техники

ПРАКТИЧЕСКАЯ РАБОТА №1

по дисциплине
«Проектирование интеллектуальных систем (Часть 1/2)»

Студент группы ИКБО-04-22 Егоров Л.А.
(Ф.И.О. студента)

Руководитель работы Холмогоров В.В.
(Ф.И.О. преподавателя)

Москва 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача поиска ассоциативных правил заключается в нахождении зависимостей в транзакциях - насколько часто один товар берут в паре с другим товаром, насколько часто одно явление происходит зависимо от второго.

В рассматриваемой предметной области «Анализ любимых исполнителей» задача поиска ассоциативных правил позволяет создать систему, позволяющую рекомендовать пользователю новых исполнителей на основе уже выбранных любимых.

1 ПОСТАНОВКА ЗАДАЧИ

Выбранная предметная область: «Анализ любимых исполнителей». В данной предметной области транзакциями являются множества любимых исполнителей у каждого пользователя.

Определена следующая цель — реализовать алгоритмы поиска ассоциативных правил в рамках поставленной предметной области.

Поставлены следующие задачи:

– определить и изучить датасет, соотвествующий выбранной предметной области;
– изучить алгоритмы поиска ассоциативных правил (Apriori, Eclat, FP-Growth);
– написать программный код для реализации указанных алгоритмов;
– сравнить основные показатели производительности алгоритмов: качество результатов, скорость работы и требуемое количество памяти.

2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Для того, чтобы выявить ассоциативные правила, в первую очередь необходимо понимать, как определить качество этих правил. Для этого можно ввести следующие метрики (далее A — множество объектов, составляющих условие, C — множество объектов, составляющих следствие, $A \to C$ — рассматриваемся ассоциация, T — множество всех транзакций):

1. **Поддержка**

Показывает вероятность встретить набор A и C в одной транзакции одновременно и высчитывается по Формуле 2.1.

$$supp(A \to C) = \frac{|A \cup C|}{|T|} \tag{2.1}$$

2. **Доверие**

Показывает, насколько часто правило оказывается верным, т.е. по сути является условной вероятностью, что в наборе окажется множество C, если в нём есть множество A. Вычисляется по Формуле 2.2.

$$\text{conf}(A \to C) = \frac{\text{supp}(A \to C)}{\text{supp}(A)} \tag{2.2}$$

3. **Достоверность (убеждённость)**

Показывает независимость A и C - чем больше значение, тем более зависимы. Вычисляется по Формуле 2.3.

$$\text{conv}(A \to C) = \frac{1 - \text{sup}(C)}{1 - \text{conf}(A \to C)} \tag{2.3}$$

4. **Лифт**

Также показывает, насколько независимы A и C друг от друга. Если лифт равен 1, то условие и следствие независимы, если больше 1, то зависимы, а если меньше 1, то наличие множества A в наборе имеет отрицательный эффект на присутствие второго объекта, и наоборот. Вычисляется по Формуле 2.4.

$$\text{lift}(A \to C) = \frac{\text{supp}(A \cup C)}{\text{supp}(A) \times \text{supp}(C)} \tag{2.4}$$

5. **Рычаг**

Также показывает, насколько независимы A и C друг от друга. Вычисляется как разность вероятности встретить A и C одновременно и мат.ожидания A и C, как если бы они были независимы друг от друга (Формула 2.5).

$$\text{leve}(A \to C) = \text{supp}(A \cup C) - \text{supp}(A) \times \text{supp}(C) \qquad (2.5)$$

Алгоритмы поиска ассоциативных правил рассчитаны на ускорение перебора всех возможных наборов условий и следствий за счёт отбрасывания заведомо редких наборов. Поэтому наиболее применимой метрикой является поддержка, показывающая вероятность встретить набор во всём множестве транзакций.

2.1 Алгоритм Apriori

Для данного алгоритма данные представляются в нормализованном виде, т.е. если в i-ой транзакции элемент *Item* встречается, то в датасете в соответствующей ячейке будет стоять 1, и 0 в обратном случае.

Далее строится префиксное дерево - родителем каждого узла является набор, содержащий префикс данного набора без одного элемента. Для первого уровня просто отбираются те элементы, которые встречаются чаще заданного порога, далее каждый уровень заполняется, используя наборы из предыдущего.

Для генерации наборов k-го уровня объединяются попарно все наборы k - 1 уровня, которые имеют одинаковый префикс. Для полученного набора проверяется множество его подмножеств, состоящих из k - 1 элемента - если хотя бы одно такое подмножество встречается реже заданного порога, то набор отбрасывается.

Алгоритм выполняется до тех пор, пока есть возможность составлять набор более высоких уровней.

2.2 Алгоритм Eclat

В основе данного алгоритма также лежит префиксное дерево, однако отличием от Apriori алгоритма является способ проверки частотности набора. Перед началом алгоритма для каждого уникального элемента составляется

множество транзакций, в котором он встречается, затем отбрасываются те, которые встречают реже заданного порога.

Далее для составления набора более высокого уровня берутся попарно наборы с одинаковыми префиксами, и при объединении наборов пересекаются множества транзакций, в которых они встречаются. Таким образом, получается новое множество транзакций, и если его размер меньше порога, то набор отбрасывается.

Алгоритм выполняется до тех пор, пока есть возможность составлять набор более высоких уровней.

2.3 Алгоритм FP-Growth

Алгоритм предполагает иной подход - вместо генерации кандидатов происходит сохранение всего множества транзакций в виде дерева, а затем с помощью обхода этого дерева выявляются часто встречающиеся наборы.

Структура узла данного дерева:

– значение;

– количество повторений.

Для составления дерева наборы объектов в каждой транзакции сортируются по убыванию частотности объектов во всём множестве транзакций. Затем инициализируется дерево с пустым корнем. Далее осуществляется проход по всем транзакциям, где для каждой транзакции выполняются следующие действия:

1. Если элемент не входит в список детей текущего узла, то он добавляется в него, иначе для подходящего узла увеличивается счётчик количества повторений на единицу
2. Выполняется переход к полученному узлу

Таким образом, на верхнем уровне дерева расположатся самые частые элементы, а листьями будут самые редкие.

Проход по дереву осуществляется по элементам в порядке возрастания их частотности. Для каждого элемента вычисляется, насколько часто

другие элементы являются его предками (т.е. родителями, прародителями, прапрародителями…). После подсчёта в новый набор идут сам рассматриваемый элемент и те элементы, частота которых выше установленного порога.

2.4 Генерация ассоциативных правил

Все рассмотренные выше алгоритмы предназначены для ускорения поиска часто встречающихся наборов, но генерация самих правил для них общая.

Из каждого множества размером k рассматриваются все подмножества размером k - 1 — эти подмножества становятся условием в правиле, а оставшийся элемент — следствием. Уже для составленных правил выполняется отбор на основе, например, метрики доверия.

3 ОПИСАНИЕ ДАННЫХ

3.1 Генерация данных

Ввиду отсутствия в общем доступе информации от стриминговых сервисов о том, какие любимые исполнители есть у пользователей (или ввиду отсутствия у автора данного отчёта навыков по поиску такого датасета) решено сгенерировать искусственные данные.

Для этого выбрано 10 исполнителей из двух разных жанров (Таблица 3.1).

Таблица 3.1 — Выбранные исполнители

Рок	**Хип-хоп**
Queen	Kendrick Lamar
Scorpions	Kanye West
Led Zeppelin	Tyler, the Creator
Black Sabbath	JAY-Z
Deep Purple	2Pac

В каждом столбце элементы отсортированы по убыванию количества слушателей за месяц на стриминговом сервисе Spotify.

Далее определено две эвристики:

1. Пользователь более вероятно будет слушать исполнителей из одного жанра, чем из разных.
2. Более популярные исполнители должны чаще встречаться в базе транзакций.

На их основе составлена матрица вероятностей (Рисунок 3.1), где в каждой столбце указано число, показывающее, с какой вероятностью следующим исполнителем будет исполнитель на указанной строке - чем больше это число, тем выше вероятность. Сама матрица не нормализована, однако при генерации данных все вероятности автоматически нормализуются, чтобы быть в интервале $[0, 1]$.

	Queen	Scorpions	Led Zeppelin	Black Sabbath	Deep Purple	Kendrick Lamar	Kanye West	Tyler, the Creator	Jay-Z	2pac
Queen	0	15	15	15	15	5	5	5	5	5
Scorpions	12	0	12	12	12	4	4	4	4	4
Led Zeppelin	9	9	0	9	9	3	3	3	3	3
Black Sabbath	6	6	6	0	6	2	2	2	2	2
Deep Purple	3	3	3	3	0	1	1	1	1	1
Kendrick Lamar	5	5	5	5	5	0	15	15	15	15
Kanye West	4	4	4	4	4	12	0	12	12	12
Tyler, the Creator	3	3	3	3	3	9	9	0	9	9
Jay-Z	2	2	2	2	2	6	6	6	0	6
2pac	1	1	1	1	1	3	3	3	3	0

Рисунок 3.1 — Матрица вероятностей перехода к следующему исполнителю

Сама генерация состоит из нескольких этапов:

1. Псевдослучайным образом выбирается количество элементов в транзакции.
2. Псевдослучайным образом выбирается первый элемент транзакции.
3. На основе указанных выше вероятностей выбираются следующие элементы.

Код генерации представлен в Листинге А.1.

4 ПРАКТИЧЕСКАЯ ЧАСТЬ

Для всех алгоритмов задан одинаковый минимальный порог поддержки: 4.

4.1 Реализация алгоритма Apriori

Для алгоритма Apriori множество транзакций представлено в нормализованном виде (Рисунок 4.1).

100%|██████████| 50000/50000 [00:12<00:00, 3969.87it/s]

	queen	kendrick lamar	black sabbath	led zeppelin	tyler, the creator	jay-z	scorpions	deep purple	kanye west	2pac
0	1	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
2	0	1	0	1	0	0	0	0	0	0
3	1	0	1	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
49995	0	0	0	1	0	0	1	0	0	0
49996	0	0	0	0	0	0	1	0	0	0
49997	0	0	0	1	0	0	0	0	0	0
49998	0	1	0	1	0	0	0	0	0	1
49999	0	1	0	0	1	0	0	0	1	0

50000 rows × 10 columns

Рисунок 4.1 — Нормализованная матрица

После выполнения самого алгоритма получено много частых наборов, которые затем были переведены в правила (Рисунок 4.2).

	Условие	Следствие	Поддержка	Достоверность	Убеждённость	Лифт	Рычаг
0	[kendrick lamar]	[queen]	8.358	23.422262	83.755412	0.653122	-0.044390
1	[queen]	[kendrick lamar]	8.358	23.306006	83.860544	0.653122	-0.044390
2	[black sabbath]	[queen]	7.298	43.183432	112.886086	1.204156	0.012373
3	[queen]	[black sabbath]	7.298	20.350231	104.331753	1.204156	0.012373
4	[led zeppelin]	[queen]	10.376	43.428763	113.375636	1.210997	0.018079
...	...	...	...	...	...	...	...
109	[kendrick lamar, queen]	[kanye west]	2.150	25.723857	93.968262	0.851671	-0.003745
110	[kanye west, queen]	[kendrick lamar]	2.150	31.899110	94.442231	0.893933	-0.002551
111	[kendrick lamar, 2pac]	[queen]	0.528	14.095035	74.661575	0.393035	-0.008154
112	[kendrick lamar, queen]	[2pac]	0.528	6.317301	97.362694	0.718855	-0.002065
113	[2pac, queen]	[kendrick lamar]	0.528	29.864253	91.702168	0.836909	-0.001029

Рисунок 4.2 — Правила, полученные алгоритмом Apriori

Далее из этих правил были выбраны те, у которых значение лифта больше единицы, а значение достоверности больше 30% (Рисунок 4.3)

	Условие	Следствие	Поддержка	Достоверность	Убеждённость	Лифт	Рычаг
22	[kendrick lamar]	[tyler, the creator]	10.832	30.355341	108.628575	1.246831	0.021444
89	[2pac]	[kanye west]	2.974	33.841602	105.498323	1.120434	0.003197
75	[jay-z]	[kanye west]	6.006	35.234073	107.766541	1.166537	0.008574
79	[deep purple]	[scorpions]	3.286	35.417116	107.455715	1.157346	0.004467
40	[black sabbath]	[scorpions]	6.054	35.822485	108.134446	1.170593	0.008823
67	[tyler, the creator]	[kanye west]	8.734	35.874476	108.842776	1.187739	0.013805
52	[led zeppelin]	[scorpions]	8.732	36.547798	109.370516	1.194294	0.014206
30	[kendrick lamar]	[kanye west]	13.276	37.204349	111.147825	1.231769	0.024980
11	[queen]	[scorpions]	13.514	37.683342	111.363481	1.231401	0.025395
12	[deep purple]	[queen]	3.906	42.099590	110.772964	1.173933	0.005787
33	[2pac]	[kendrick lamar]	3.746	42.626309	112.100160	1.194550	0.006101
25	[jay-z]	[kendrick lamar]	7.268	42.637569	112.122166	1.194865	0.011853
2	[black sabbath]	[queen]	7.298	43.183432	112.886086	1.204156	0.012373
4	[led zeppelin]	[queen]	10.376	43.428763	113.375636	1.210997	0.018079
31	[kanye west]	[kendrick lamar]	13.276	43.954443	114.756644	1.231769	0.024980
10	[scorpions]	[queen]	13.514	44.160512	114.861369	1.231401	0.025395
23	[tyler, the creator]	[kendrick lamar]	10.832	44.491908	115.867792	1.246831	0.021444

Рисунок 4.3 — Отфильтрованные правила

Код реализации алгоритма представлен в Листинге Б.1.

4.2 Реализация алгоритма Eclat

Для алгоритма Eclat для каждого элемента составлено множество транзакций (Рисунок 4.4).

	Item	Transactions
0	[2pac]	[19, 46, 48, 52, 53, 58, 83, 84, 90, 93, 109, ...
1	[black sabbath]	[1, 3, 6, 7, 8, 12, 13, 23, 28, 30, 31, 34, 38...
2	[deep purple]	[9, 19, 22, 24, 26, 27, 32, 40, 50, 55, 73, 78...
3	[jay-z]	[7, 27, 28, 54, 56, 59, 60, 64, 76, 77, 80, 84...
4	[kanye west]	[11, 18, 20, 26, 35, 39, 46, 47, 52, 56, 58, 6...
5	[kendrick lamar]	[0, 2, 4, 5, 6, 11, 15, 19, 20, 21, 29, 33, 34...
6	[led zeppelin]	[2, 11, 13, 14, 15, 16, 17, 21, 24, 27, 29, 31...
7	[queen]	[0, 3, 7, 9, 11, 15, 16, 17, 19, 21, 23, 25, 3...
8	[scorpions]	[8, 10, 16, 21, 22, 24, 25, 29, 30, 31, 32, 36...
9	[tyler, the creator]	[5, 18, 24, 25, 29, 39, 45, 54, 61, 62, 63, 64...

Рисунок 4.4 — Изменённый вид датасета

После выполнения самого алгоритма получено много частых наборов, которые затем были переведены в правила (Рисунок 4.5).

	Условие	Следствие	Поддержка	Достоверность	Убеждённость	Лифт	Рычаг
0	[2pac]	[black sabbath]	0.800	9.103323	91.422484	0.538658	-0.006852
1	[black sabbath]	[2pac]	0.800	4.733728	95.744273	0.538658	-0.006852
2	[2pac]	[deep purple]	0.432	4.915794	95.412271	0.529833	-0.003834
3	[deep purple]	[2pac]	0.432	4.656176	95.666396	0.529833	-0.003834
4	[2pac]	[jay-z]	1.682	19.139736	102.589326	1.122829	0.001840
...	...	...	...	...	...	...	...
445	[led zeppelin, tyler, the creator]	[scorpions]	0.842	24.970344	92.494093	0.815971	-0.001899
446	[scorpions, tyler, the creator]	[led zeppelin]	0.842	18.921348	93.869346	0.791953	-0.002212
447	[queen, scorpions]	[tyler, the creator]	1.320	9.767648	83.843542	0.401201	-0.019701
448	[queen, tyler, the creator]	[scorpions]	1.320	24.645258	92.095066	0.805348	-0.003190
449	[scorpions, tyler, the creator]	[queen]	1.320	29.662921	91.186613	0.827141	-0.002759

Рисунок 4.5 — Правила, полученные алгоритмом Eclat

Далее из этих правил были выбраны те, у которых значение лифта больше единицы, а значение достоверности больше 30% (Рисунок 4.6 - 4.7).

	Условие	Следствие	Поддержка	Достоверность	Убеждённость	Лифт	Рычаг
76	[kendrick lamar]	[tyler, the creator]	10.832	30.355341	108.628575	1.246831	0.021444
399	[kanye west, kendrick lamar]	[tyler, the creator]	4.070	30.656824	109.100859	1.259214	0.008378
166	[2pac, tyler, the creator]	[kanye west]	0.818	33.225020	104.524180	1.100021	0.000744
154	[2pac, kendrick lamar]	[kanye west]	1.264	33.742659	105.340780	1.117159	0.001326
6	[2pac]	[kanye west]	2.974	33.841602	105.498323	1.120434	0.003197
330	[deep purple, led zeppelin]	[scorpions]	0.876	34.569850	106.064253	1.129660	0.001005
267	[black sabbath, led zeppelin]	[scorpions]	1.580	34.801762	106.441527	1.137238	0.001907
213	[black sabbath, deep purple]	[scorpions]	0.610	34.817352	106.466984	1.137748	0.000739
358	[jay-z, tyler, the creator]	[kanye west]	1.634	35.034305	107.435163	1.159923	0.002253
48	[jay-z]	[kanye west]	6.006	35.234073	107.766541	1.166537	0.008574
44	[deep purple]	[scorpions]	3.286	35.417116	107.455715	1.157346	0.004467
30	[black sabbath]	[scorpions]	6.054	35.822485	108.134446	1.170593	0.008823
346	[jay-z, kendrick lamar]	[kanye west]	2.606	35.855806	108.811096	1.187121	0.004108
69	[tyler, the creator]	[kanye west]	8.734	35.874476	108.842776	1.187739	0.013805
336	[deep purple, queen]	[scorpions]	1.408	36.047107	108.514247	1.177933	0.002127
80	[led zeppelin]	[scorpions]	8.732	36.547798	109.370516	1.194294	0.014206

Рисунок 4.6 — Отфильтрованные правила

80	[led zeppelin]	[scorpions]	8.732	36.547798	109.370516	1.194294	0.014206
61	[kendrick lamar]	[kanye west]	13.276	37.204349	111.147825	1.231769	0.024980
273	[black sabbath, queen]	[scorpions]	2.722	37.297890	110.678891	1.218806	0.004887
401	[kendrick lamar, tyler, the creator]	[kanye west]	4.070	37.573855	111.805719	1.244003	0.007983
84	[queen]	[scorpions]	13.514	37.683342	111.363481	1.231401	0.025395
438	[led zeppelin, queen]	[scorpions]	3.972	38.280648	112.441232	1.250920	0.007967
138	[2pac, jay-z]	[kendrick lamar]	0.664	39.476813	106.266711	1.106289	0.000638
210	[black sabbath, deep purple]	[queen]	0.700	39.954338	106.815376	1.114113	0.000717
42	[deep purple]	[queen]	3.906	42.099590	110.772964	1.173933	0.005787
264	[black sabbath, led zeppelin]	[queen]	1.920	42.290749	111.139893	1.179264	0.002919
327	[deep purple, led zeppelin]	[queen]	1.076	42.462510	111.471668	1.184053	0.001673
153	[2pac, kanye west]	[kendrick lamar]	1.264	42.501681	111.857184	1.191057	0.002028
8	[2pac]	[kendrick lamar]	3.746	42.626309	112.100160	1.194550	0.006101
50	[jay-z]	[kendrick lamar]	7.268	42.637569	112.122166	1.194865	0.011853
337	[deep purple, scorpions]	[queen]	1.408	42.848448	112.224424	1.194815	0.002296
178	[2pac, tyler, the creator]	[kendrick lamar]	1.060	43.054427	112.942933	1.206547	0.001815
370	[jay-z, tyler, the creator]	[kendrick lamar]	2.010	43.096055	113.025555	1.207714	0.003457
28	[black sabbath]	[queen]	7.298	43.183432	112.886086	1.204156	0.012373
345	[jay-z, kanye west]	[kendrick lamar]	2.606	43.389943	113.612322	1.215950	0.004628
78	[led zeppelin]	[queen]	10.376	43.428763	113.375636	1.210997	0.018079
60	[kanye west]	[kendrick lamar]	13.276	43.954443	114.756644	1.231769	0.024980
85	[scorpions]	[queen]	13.514	44.160512	114.861369	1.231401	0.025395
77	[tyler, the creator]	[kendrick lamar]	10.832	44.491908	115.867792	1.246831	0.021444
274	[black sabbath, scorpions]	[queen]	2.722	44.962009	116.534049	1.253751	0.005509
439	[led zeppelin, scorpions]	[queen]	3.972	45.487861	117.658197	1.268414	0.008405
400	[kanye west, tyler, the creator]	[kendrick lamar]	4.070	46.599496	120.440811	1.305893	0.009534

Рисунок 4.7 — Отфильтрованные правила

Код реализации алгоритма представлен в Листинге В.1.

4.3 Реализация алгоритма FP-Growth

В начале элементы в каждой транзакции отсортированы по убыванию частотности (Рисунок 4.8)

Рисунок 4.8 — Время выполнения сортировки элементов

Затем выполнено построение дерева из элементов (Рисунок 4.9) — дерево представлено строками из элементов, где записан сам узел и список его детей (узел None соответствует пустому корневому узлу).

```
0(None: 0) -> [1(queen: 17931)  3(black sabbath: 974)  4(kendrick lamar: 13663)  11(scorpions: 6160)  16(led zepp
1(queen: 17931) -> [2(kendrick lamar: 4179)  6(black sabbath: 784)  9(jay-z: 312)  13(deep purple: 323)  19(scorp
2(kendrick lamar: 4179) -> [14(kanye west: 936)  18(led zeppelin: 438)  24(deep purple: 92)  27(scorpions: 1108)
14(kanye west: 936) -> [15(led zeppelin: 76)  59(2pac: 52)  92(tyler, the creator: 187)  125(black sabbath: 61)
15(led zeppelin: 76) -> []
59(2pac: 52) -> []
92(tyler, the creator: 187) -> []
125(black sabbath: 61) -> []
197(deep purple: 33) -> []
199(jay-z: 121) -> []
18(led zeppelin: 438) -> [134(2pac: 16)  200(black sabbath: 82)  278(deep purple: 35)  319(jay-z: 39)]
134(2pac: 16) -> []
200(black sabbath: 82) -> []
278(deep purple: 35) -> []
319(jay-z: 39) -> []
24(deep purple: 92) -> [25(2pac: 7)]
25(2pac: 7) -> []
27(scorpions: 1108) -> [28(led zeppelin: 184)  50(black sabbath: 96)  176(kanye west: 139)  209(deep purple: 72)
28(led zeppelin: 184) -> []
50(black sabbath: 96) -> []
176(kanye west: 139) -> []
209(deep purple: 72) -> []
235(tyler, the creator: 95) -> []
256(jay-z: 53) -> []
312(2pac: 23) -> []
...
93(2pac: 421) -> []
103(deep purple: 447) -> [126(2pac: 20)]
126(2pac: 20) -> []
```

Рисунок 4.9 — Построенное дерево

После выполнения самого алгоритма получено много частых наборов, которые затем были переведены в правила (Рисунок 4.10).

	Условие	Следствие	Поддержка	Достоверность	Убеждённость	Лифт	Рычаг
0	[2pac]	[queen]	1.768	20.118343	80.291274	0.560993	-0.013836
1	[queen]	[2pac]	1.768	4.930009	95.941947	0.560993	-0.013836
2	[2pac]	[kendrick lamar]	3.746	42.626309	112.100160	1.194550	0.006101
3	[kendrick lamar]	[2pac]	3.746	10.497702	101.910233	1.194550	0.006101
4	[2pac]	[black sabbath]	0.800	9.103323	91.422484	0.538658	-0.006852
...	...	...	...	...	...	...	...
2521	[scorpions, queen]	[kendrick lamar]	2.216	16.397810	76.930999	0.459528	-0.026063
2522	[scorpions, kendrick lamar]	[queen]	2.216	31.729668	93.947104	0.884771	-0.002886
2523	[queen, kendrick lamar]	[scorpions]	2.216	26.513520	94.436419	0.866398	-0.003417
2524	[kendrick lamar]	[queen]	8.358	23.422262	83.755412	0.653122	-0.044390
2525	[queen]	[kendrick lamar]	8.358	23.306006	83.860544	0.653122	-0.044390

2526 rows × 7 columns

Рисунок 4.10 — Правила, полученные алгоритмом FP-Growth

Далее из этих правил были выбраны те, у которых значение лифта больше единицы, а значение достоверности больше 30% (Рисунок 4.11).

	Условие	Следствие	Поддержка	Достоверность	Убеждённость	Лифт	Рычаг
2454	[kendrick lamar]	[tyler, the creator]	10.832	30.355341	108.628575	1.246831	0.021444
2473	[kendrick lamar, kanye west]	[tyler, the creator]	4.070	30.656824	109.100859	1.259214	0.008378
105	[2pac, tyler, the creator]	[kanye west]	0.818	33.225020	104.524180	1.100021	0.000744
60	[2pac, kendrick lamar]	[kanye west]	1.264	33.742659	105.340780	1.117159	0.001326
16	[2pac]	[kanye west]	2.974	33.841602	105.498323	1.120434	0.003197
1149	[deep purple, led zeppelin]	[scorpions]	0.876	34.569850	106.064253	1.129660	0.001005
1795	[black sabbath, led zeppelin]	[scorpions]	1.580	34.801762	106.441527	1.137238	0.001907
1137	[deep purple, black sabbath]	[scorpions]	0.610	34.817352	106.466984	1.137748	0.000739
2185	[jay-z, tyler, the creator]	[kanye west]	1.634	35.034305	107.435163	1.159923	0.002253
2144	[jay-z]	[kanye west]	6.006	35.234073	107.766541	1.166537	0.008574
1085	[deep purple]	[scorpions]	3.286	35.417116	107.455715	1.157346	0.004467
1752	[black sabbath]	[scorpions]	6.054	35.822485	108.134446	1.170593	0.008823
2170	[jay-z, kendrick lamar]	[kanye west]	2.606	35.855806	108.811096	1.187121	0.004108
2457	[tyler, the creator]	[kanye west]	8.734	35.874476	108.842776	1.187739	0.013805
1104	[deep purple, queen]	[scorpions]	1.408	36.047107	108.514247	1.177933	0.002127
2352	[led zeppelin]	[scorpions]	8.732	36.547798	109.370516	1.194294	0.014206
371	[2pac, black sabbath, deep purple]	[scorpions]	0.020	37.037037	110.220353	1.210282	0.000035
2501	[kendrick lamar]	[kanye west]	13.276	37.204349	111.147825	1.231769	0.024980
1768	[black sabbath, queen]	[scorpions]	2.722	37.297890	110.678891	1.218806	0.004887
2471	[tyler, the creator, kendrick lamar]	[kanye west]	4.070	37.573855	111.805719	1.244003	0.007983
2518	[queen]	[scorpions]	13.514	37.683342	111.363481	1.231401	0.025395
2362	[led zeppelin, queen]	[scorpions]	3.972	38.280648	112.441232	1.250920	0.007967
52	[2pac, jay-z]	[kendrick lamar]	0.664	39.476813	106.266711	1.106289	0.000638
1093	[deep purple, black sabbath]	[queen]	0.700	39.954338	106.815376	1.114113	0.000717
1073	[deep purple]	[queen]	3.906	42.099590	110.772964	1.173933	0.005787
1760	[black sabbath, led zeppelin]	[queen]	1.920	42.290749	111.139893	1.179264	0.002919
1096	[deep purple, led zeppelin]	[queen]	1.076	42.462510	111.471668	1.184053	0.001673
61	[2pac, kanye west]	[kendrick lamar]	1.264	42.501681	111.857184	1.191057	0.002028
2	[2pac]	[kendrick lamar]	3.746	42.626309	112.100160	1.194550	0.006101
2136	[jay-z]	[kendrick lamar]	7.268	42.637569	112.122166	1.194865	0.011853
1105	[deep purple, scorpions]	[queen]	1.408	42.848448	112.224424	1.194815	0.002296
49	[2pac, tyler, the creator]	[kendrick lamar]	1.060	43.054427	112.942933	1.206547	0.001815
2165	[jay-z, tyler, the creator]	[kendrick lamar]	2.010	43.096055	113.025555	1.207714	0.003457
1742	[black sabbath]	[queen]	7.298	43.183432	112.886086	1.204156	0.012373
2171	[jay-z, kanye west]	[kendrick lamar]	2.606	43.389943	113.612322	1.215950	0.004628
2346	[led zeppelin]	[queen]	10.376	43.428763	113.375636	1.210997	0.018079
2500	[kanye west]	[kendrick lamar]	13.276	43.954443	114.756644	1.231769	0.024980
2517	[scorpions]	[queen]	13.514	44.160512	114.861369	1.231401	0.025395
2453	[tyler, the creator]	[kendrick lamar]	10.832	44.491908	115.867792	1.246831	0.021444
1769	[black sabbath, scorpions]	[queen]	2.722	44.962009	116.534049	1.253751	0.005509
2363	[led zeppelin, scorpions]	[queen]	3.972	45.487861	117.658197	1.268414	0.008405
2472	[tyler, the creator, kanye west]	[kendrick lamar]	4.070	46.599496	120.440811	1.305893	0.009534

Рисунок 4.11 — Отфильтрованные правила

Код реализации алгоритма представлен в Листинге Г.1.

ЗАКЛЮЧЕНИЕ

В результате реализации алгоритмов выяснено, что самым быстрым алгоритмом является Eclat, поскольку нормализация данных под него занимает мало времени, в отличие от Apriori и FP-Growth. В то же время, FP_Growth является самым экономным в плане памяти за счёт того, что единственной структурой, которую нужно держать в памяти, является легковесное дерево, в отличие от массивов и множеств в Apriori и Eclat. Также FP-Growth выдал наибольшее количество результатов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ассоциативные правила, или пиво с подгузниками [Электронный ресурс]: Habr. URL: https://habr.com/ru/companies/ods/articles/353502/.
2. Azevedo, P. J., Jorge, A. M. Comparing Rule Measures for Predictive Association Rules 2007.

ПРИЛОЖЕНИЯ

Приложение А — Генерация данных.

Приложение Б — Реализация алгоритма Apriori.

Приложение В — Реализация алгоритма Eclat.

Приложение Г — Реализация алгоритма FP-Growth.

Приложение А

Генерация данных

Листинг А.1 — Генерация данных

```
groups = ['Queen', 'Scorpions', 'Led Zeppelin', 'Black Sabbath', 'Deep Purple',
          'Kendrick Lamar', 'Kanye West', 'Tyler, the Creator', 'Jay-Z', '2pac']
group_df = pd.DataFrame(index=groups, columns=groups)
popularities = []
for i in range(len(groups) // 5):
    popularities.extend(range(5, 0, -1))


for i in range(len(groups) // 5):
    for j in range(5 * i):
        for k in range(5 * i, 5 * i + 5):
            group_df.iloc[j, k] = group_df.iloc[k, j] = 1
    for j in range(5 * i, 5 * i + 5):
        for k in range(5 * i, 5 * i + 5):
            if j != k:
                group_df.iloc[j, k] = group_df.iloc[k, j] = 3
            else:
                group_df.iloc[j, k] = 0
    for j in range(5 * i + 5, len(groups)):
        for k in range(5 * i, 5 * i + 5):
            group_df.iloc[j, k] = group_df.iloc[k, j] = 3

for i in range(len(groups)):
    group_df.iloc[i] *= popularities[i]
group_df
```

Приложение Б

Реализация алгоритма Apriori

Листинг Б.1 — Алгоритм Apriori

```
results: list[list[set]] = []
while True:
    f_k: list[set] = []
    if k == 1:
        for item in unique_items:
            if norm_df.sum()[item] >= f:
                f_k.append({item})
    else:
        if k == 2:
          combs: list[set] = [x[0].union(x[1]) for x in combinations(results[0],
r=2)]
        else:
            previous = results[k - 2]
            combs = []
            for i, elem in tqdm(enumerate(previous)):
                _set = set(sorted(elem)[:-1])
                if any(_set.issubset(x) for x in combs):
                    continue
                for j, elem1 in enumerate(previous):
                    if i == j:
                        continue
              if _set.issubset(elem1) and all(any(len(set(subset).difference(x))
== 0 for x in previous) for subset in combinations(elem.union(elem1), r=k-1)):
                        combs.append(elem.union(elem1))

        for elem in combs:
            cnt = count_rows(list(elem))
            if cnt >= f:
                f_k.append(elem)

    if len(f_k) == 0:
        break
    results.append(f_k)
    k += 1
```

Приложение В

Реализация алгоритма Eclat

Окончание Листинга В.1

```
min_support = 4
start_point = 0
k = 2
while True:
    end_point = len(eclat_df)

    new_dict = {'Item': [], 'Transactions': []}
    for i in tqdm(range(start_point, end_point)):
        for j in range(i + 1, end_point):
            item1, item2 = eclat_df.iloc[i]['Item'], eclat_df.iloc[j]['Item']
           if k > 2 and not all(x == y for x in item1[:k - 2] for y in item2[:k
- 2]):
                break

                tr1, tr2 = eclat_df.iloc[i]['Transactions'], eclat_df.iloc[j]
['Transactions']
            if len(set(item1).union(set(item2))) == k:
                new_dict['Item'].append(sorted(set(item1).union(set(item2))))

new_dict['Transactions'].append(list(set(tr1).intersection(set(tr2))))

    eclat_df = pd.concat([
        eclat_df,
        pd.DataFrame(new_dict)
    ], ignore_index=True)
    if len(eclat_df) == end_point:
        break
    k += 1
    start_point = end_point
      eclat_df = eclat_df[eclat_df['Transactions'].apply(lambda t: len(t) >=
min_support)]
```

Приложение Г

Реализация алгоритма FP-Growth

Листинг Г.1 — Алгоритм FP-Growth

```
class FPTree:
    num: int = 0

    def __init__(self, value: str | None = None):
        self.value: str | None = value
        self.cnt: int = 0
        self.children: list['FPTree'] = []
        self.num: int = FPTree.num
        FPTree.num += 1

    def add(self, value: str):
        if value not in [x.value for x in self.children]:
            node = FPTree(value)
            node.cnt += 1
            self.children.append(node)
            return node
        else:
            index = [x.value for x in self.children].index(value)
            self.children[index].cnt += 1
            return self.children[index]

    def __str__(self):
       res = f'{self.num}({self.value}: {self.cnt}) -> [' + '  '.join([f'{x.num}
({x.value}: {x.cnt})' for x in self.children]) + ']\n'
        for child in self.children:
            res += str(child)
        return res

    def find(self, value: str):
        result = dict.fromkeys(unique_items, 0)

        def dfs(node: 'FPTree'):
            cnt = 0
            for child in node.children:
                if child.value == value:
                    cnt = child.cnt
                    node.children.remove(child)
                else:
                    cnt += dfs(child)
                if node.value is not None:
                    result[node.value] += cnt
            return cnt

        while dfs(self) != 0:
            pass

        return result
```