Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе №1 по дисциплине "Проектирование рекомендательных систем"

Тема Сравнение алгоритмов поиска ассоциативных правил

Студент Варламова Е. А.

Группа ИУ7-33М

Оценка (баллы) _____

Преподаватели Быстрицкая А.Ю.

Москва — 2024 г.

СОДЕРЖАНИЕ

Введение

Цель работы – сравнение алгоритмов поиска ассоциативных правил Apriori, ECLAT и FP-Growth.

Для достижения поставленной цели потребуется:

— привести описание алгоритмов Apriori, ECLAT и FP-Growth;

— привести описание используемых для исследования данных;

— провести сравнение алгоритмов по времени работы и затратам по памяти.

1 | Аналитический раздел

1.1 Задача поиска ассоциативных правил

Правило ассоциации состоит из двух частей, предшествующей и последующей. Предшествующая задача – это элемент, находящийся в данных. А последующая – это элемент или множество элементов, которые встречаются в сочетании с предшествующей задачей. [1]

В интеллектуальном анализе данных правила ассоциации являются полезными и помогают спрогнозировать поведение клиента.

Для оценки качества полученных рекомендаций используются следующие метрики [1]:

— Поддержка – позволяет узнать, насколько часто объект встречается в БД. Определяется как количество транзакций, содержащих X к общему числу транзакций: $support(X) = \frac{|t \in T, x \in X|}{|T|}$

— Достоверность – показывает, насколько хорошим является правило для предсказания правой части,когда условие слева верно. Определяется как $confidence(A \to B) = \frac{supp(A \cup B)}{supp(A)}$

— Интерес – измеряет силу правила, сравнивая полное правило с предположенной правой частью и рассчитывается, как отношение достоверности правила к частоте появления следствия – $lift(A \to B) = \frac{supp(A \cup B)}{supp(A)supp(B)}$

— Уверенность – частотность ошибок правила: $conv(A \to B) = \frac{1 - supp(B)}{1 - conf(A \to B)}$

1.2 Apriori

Принцип работы алгоритма [2]:

1. Определение минимальной поддержки: устанавливается порог минимальной поддержки, который определяет, как часто элемент или набор элементов должен встречаться в транзакциях для того, чтобы считаться значимым.

2. Сбор данных: собираются данные о транзакциях. Транзакции могут представлять собой покупки товаров в магазине, клики на веб-сайте и т.д. Каждая транзакция состоит из множества элементов (например, товаров).

3. Генерация частых одиночных элементов: алгоритм начинает с поиска всех одиночных элементов (товаров), которые удовлетворяют минимальной поддержке. Это создаёт первый набор частых элементов.

4. Генерация кандидатов: на основе найденных частых одиночных элементов создаются кандидаты для пар (двухэлементных наборов), троек и так далее. Этот процесс повторяется, увеличивая размер набора элементов.

5. Отбрасывание нечастых кандидатов: для каждого нового набора кандидатов вычисляется их поддержка. Если поддержка не достигает установленного порога, набор отбрасывается.

6. Повторение процесса: процесс генерации кандидатов и вычисления их поддержки повторяется до тех пор, пока не останется ни одного кандидата, который бы удовлетворял минимальной поддержке.

7. Генерация ассоциативных правил: на основе частых наборов элементов создаются ассоциативные правила. Каждое правило имеет форму "если A, то B где A и B — это наборы элементов. Для каждого правила вычисляется мера уверенности (confidence), которая показывает вероятность того, что если A присутствует в транзакции, то также присутствует и B.

8. Фильтрация правил: наконец, правила могут быть отфильтрованы по уверенности или другим критериям, чтобы оставить только наиболее значимые.

1.3 ECLAT

Принцип работы алгоритма [3]:

1. Построение таблицы транзакций: каждому элементу в наборе данных присваивается уникальный идентификатор, и создается таблица, где каждая строка представляет собой список идентификаторов транзакций, содержащих этот элемент.

2. Определение минимальной поддержки: устанавливается порог минимальной поддержки, аналогично алгоритму Apriori.

3. Генерация частых наборов: для каждого элемента в таблице вычисляется поддержка. Затем алгоритм начинает генерировать частые наборы, используя пересечение списков транзакций:
   — Если есть два элемента A и B, необходимо брать их списки транзакций и находить пересечение (т.е., транзакции, которые содержат оба элемента).
   — Если пересечение превышает порог минимальной поддержки, создается новый набор A, B.
4. Повторение процесса: процесс повторяется для всех возможных комбинаций элементов до тех пор, пока не будут найдены все частые наборы.
5. Генерация ассоциативных правил: на основе найденных частых наборов создаются ассоциативные правила, аналогично тому, как это делается в Apriori.

1.4 FP-Growth

Принцип работы алгоритма [4]:

1. Сбор данных и определение минимальной поддержки: определите набор транзакций и установите порог минимальной поддержки.
2. Подсчет частоты элементов: подсчитайте, как часто каждый элемент встречается в транзакциях.
3. Построение FP-дерева: создайте FP-дерево, начиная с корня, где каждый узел представляет элемент и его частоту. Элементы добавляются в дерево по порядку их частоты (от наиболее частых к наименее частым).
4. Извлечение частых наборов: для каждого элемента (листового узла) в FP-дереве создается условное FP-дерево, включающее только транзакции, содержащие данный элемент. Этот процесс повторяется рекурсивно для каждого элемента до тех пор, пока не будут найдены все частые наборы.
5. Генерация ассоциативных правил: на основе найденных частых наборов создаются ассоциативные правила.

2 | Конструкторский раздел

В данном разделе описаны данные, анализируемые в данной работе.

2.1 Market Basket Optimisation

В качестве источника данных был взят датасет, располагающийся в свободном доступе на веб-сайте Kaggle [5]. Набор данных включает в себя корзины потребителя некоторого продуктового магазина. В качестве предобработки была построена база данных транзакций, которая структурно изменялась по требованию входных данных используемых алгоритмов.

3 | Технологический раздел

В данном разделе описываются средства разработки программного обеспечения.

3.1 Средства реализации

В качестве используемого был выбран язык программирования Python [6].
Данный выбор обусловлен следующими факторами:

— Большое количество исчерпывающей документации;

— Широкий выбор доступных библиотек для разработки;

— Простота синтаксиса языка и высокая скорость разработки.

При написании программного продукта использовалась среда разработки Visual Studio Code. Данный выбор обусловлен тем, что данная среда распространяется по свободной лицензии, поставляется для конечного пользователя с открытым исходным кодом, а также имеет большое число расширений, ускоряющих разработку.

3.2 Библиотеки

При анализе и обработке датасета, а также для решения поставленных задач использовались библиотеки:

— pandas;

— numpy;

— matplotlib;

— apyory [2];

— pyECLAT [3];

— fpgrowth-py [7].

4 | Исследовательский раздел

4.1 Условия исследований

Исследование проводилось на персональном вычислительной машине со следующими характеристиками:

— процессор Intel Core i5

— операционная система Mac OS Big Sur,

— 8 Гб оперативной памяти.

Временные затраты определялись с использованием библиотеки time. Затраты по памяти определялись с использованием библиотеки memory_profiler.

В данном исследовании значение параметра минимальной поддержки изменялось от значения 0.001 до 0.01 с шагом 0.001 и от 0.01 до 0.25 с шагом 0.01. Значение минимального и максимального количества элементов было равно 2 для ECLAT, для остальных не ограничено сверху.

4.2 Зависимость времени исполнения от параметра минимальной поддержки

На рисунке 4.1 представлен график зависимости времени исполнения алгоритмов от заданного параметра минимальной поддержки.

4.3 Зависимость количества занятой памяти процессом во время исполнения алгоритмов от заданного параметра минимальной поддержки

На рисунке 4.2 представлен график зависимости количества занятой памяти процессом во время исполнения алгоритмов от заданного параметра минимальной поддержки.

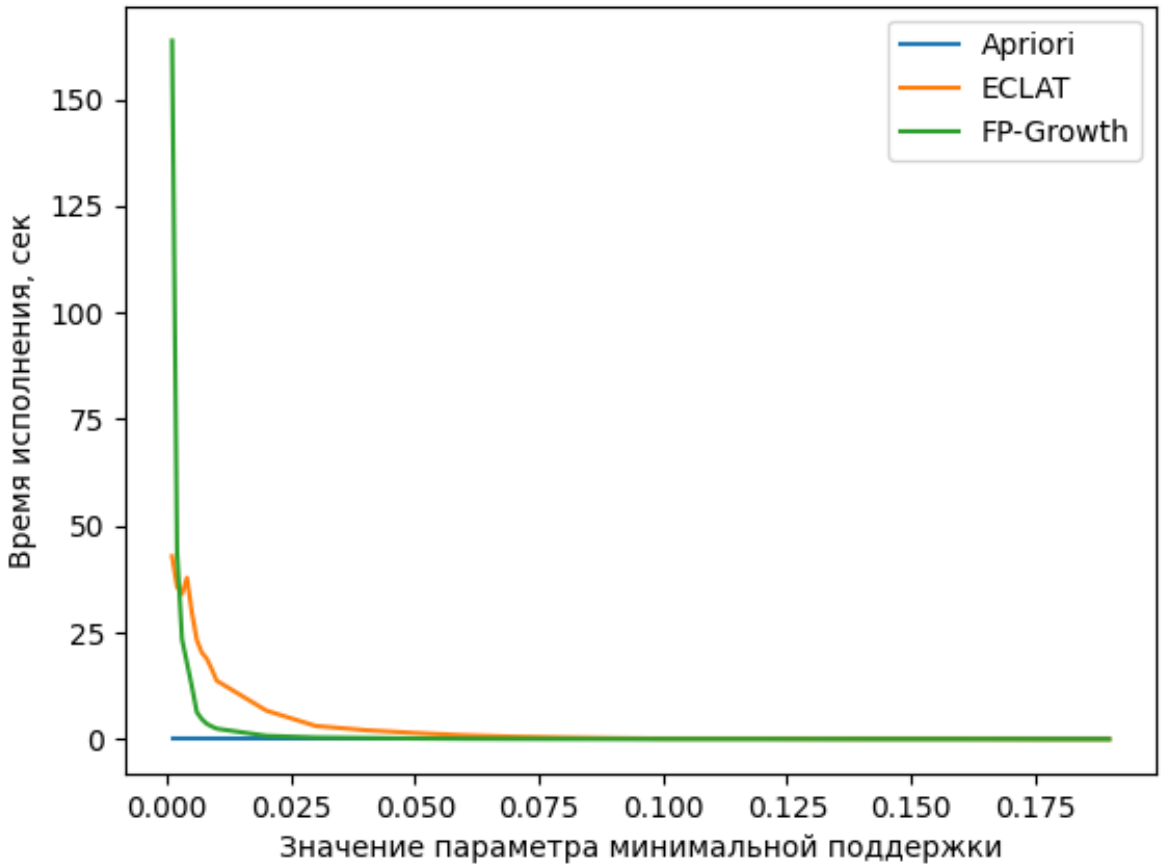

Рис. 4.1: График зависимости времени исполнения от заданного параметра минимальной поддержки.

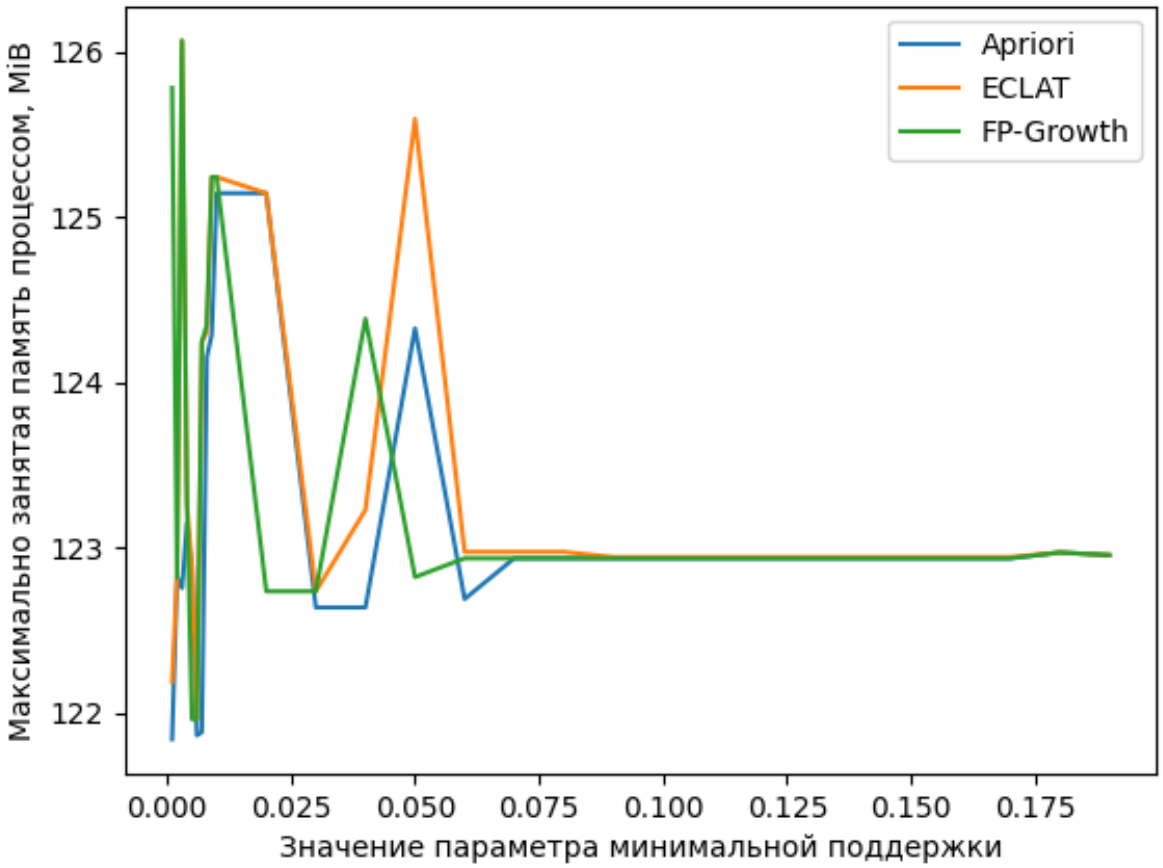

Рис. 4.2: График зависимости количества занятой процессом во время исполнения алгоритмов от заданного параметра минимальной поддержки.

Вывод

В результате проведенных исследований заметно, что по времени исполнения на меньших значениях параметра минимальной поддержки самое долгое время исполнения у алгоритма FP-Growth. Самым быстрым временем исполнения обладает алгоритм Apriori. В дальнейшем, при увеличении параметра, разница во времени исполнения у трех алгоритмов становится уже не такой заметной и при значении параметра минимальной поддержки равным 0.10 уже практически отсутствует.

Также стоит отметить, что размер задействованной памяти, потребовавшейся для исполнения каждого из алгоритмов, практически одинаков. Отличия присутствуют лишь при начальных значениях величины параметра минимальной поддержки.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было проведено сравнение алгоритмов поиска ассоциативных правил Apriori, ECLAT и FP-Growth.

Исследования показали, что при заданных условиях отличия во времени исполнения между алгоритмами не разительны, однако на меньших значениях параметра минимальной поддержки наилучшим образом себя показал именно алгоритм Apriori.

Были решены следующие задачи:

— приведено описание алгоритмов Apriori, ECLAT и FP-Growth;

— приведено описание используемых для исследования данных;

— проведено сравнение алгоритмов по времени работы и затратам по памяти.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Анатольевич О. И.* Сравнение алгоритмов построения ассоциативных правил на основе набора данных покупательских транзакций // Известия Самарского научного центра РАН. — 2018. — № 6—2.
2. Apyory: official PyPl project page [Электронный ресурс]. — Режим доступа: `https://pypi.org/project/apyori/` (дата обращения 16.09.2023).
3. pyECLAT: official PyPl project page [Электронный ресурс]. — Режим доступа: `https://pypi.org/project/pyECLAT/` (дата обращения 16.09.2023).
4. *Jiawei Han Hong Cheng D. X.* Frequent pattern mining: current status and future directions // Режим доступа: `https://sites.cs.ucsb.edu/~xyan/papers/dmkd07_frequentpattern.pdf` (дата обращения 20.09.2023). — 2006.
5. Kaggle Market Basket Optimisation Dataset [Электронный ресурс]. — Режим доступа: `https://www.kaggle.com/datasets/devchauhan1/market-basket-optimisationcsv` (дата обращения 16.09.2023).
6. Python official page [Электронный ресурс]. — Режим доступа: `https://www.python.org/` (дата обращения 10.05.2023).
7. fpgrowth-py: official PyPl project page [Электронный ресурс]. — Режим доступа: `https://pypi.org/project/fpgrowth-py/` (дата обращения 16.09.2023).