Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем

Группа 23.Б09-мм

Реализация валидации ассоциативных правил в Desbordante

Гараев Тагир Ильгизович

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
ассистент кафедры информационно-аналитических систем, Чернышев Г. А.

Санкт-Петербург
2025

Оглавление

Введение

Desbordante [2] — это высокоэффективный наукоёмкий профилировщик данных. Наукоёмкое профилирование ориентировано на выявление примитивов — формальных представлений шаблонов в данных. Примитив определяет класс зависимостей (например, функциональных), или иные объекты, такие как запрещающие ограничения и ассоциативные правила.

Ассоциативные правила (Association Rules) широко применяются в качестве примитива профилирования. Они выявляются в [1] огромных объемах данных о продажах, называемых рыночной корзиной. Запись в таких данных обычно состоит из транзакций и товаров, купленных в транзакции. Сами правила могут иметь такой вид [1]: «98% клиентов, которые покупают шины и автомобильные аксессуары, также получают услуги по обслуживанию автомобилей». Эти закономерности применяются [4] для анализа потребительского поведения (вроде поиска связей между покупками), обработки данных веб-сервисов, биоинформатических исследований, а также в задачах классификации и кластеризации данных.

На данный момент в системе Desbordante содержатся алгоритмы, способные выявлять и проверять различные примитивы в данных, в том числе алгоритм поиска ассоциативных правил Apriori [9]. Однако возможность проверять заданные пользователем правила не представлена. В данной работе будет описано добавление в проект необходимой функциональности.

1 Постановка задачи

Целью работы является реализация и внедрение алгоритма валидации ассоциативных правил. Для её достижения были поставлены следующие задачи:

- Провести обзор предметной области и задокументировать его;
- Разработать алгоритм валидации ассоциативных правил и предложить способ анализа исключений;
- Создать набор модульных тестов, проверяющих корректность выполнения алгоритма;
- Реализовать привязки для Python с помощью библиотеки pybind11 [6], привести пример использования алгоритма в Python;
- Провести первичное экспериментальное исследование производительности.

2 Обзор

2.1 Обзор существующих решений

Алгоритмы валидации проверяют, удерживается ли определённый инстанс примитива — конкретное правило данного типа на определённом наборе данных. Пользователь формирует гипотезу о данных, затем проверяет её корректность с помощью одного из алгоритмов валидации. Если же гипотеза не выполняется, в зависимости от примитива система может указать на конкретные места в данных с аномалиями или предоставить параметр ошибки.

На данный момент в библиотеке Desbordante реализовано множество алгоритмов валидации, например:

- Алгоритм для эффективной верификации особого вида зависимостей в данных — запрещающих ограничений [7];
- Алгоритм валидации вероятностных функциональных зависимостей [8];
- Алгоритм валидации графовых функциональных зависимостей, реализованный Антоном Черниковым на основе статьи [3].

В рамках ассоциативных правил задача валидации заключается в проверке поддержки и доверия правила для конкретного набора данных.

2.2 Основные понятия

Определим ключевые понятия, которые будут часто встречаться далее. Все последующие определения приводятся по источнику [4].

Определение 1. *Пусть* $I = \{i_1, i_2, \ldots, i_n\}$ *— множество некоторых сущностей, называемых* ***предметами***. ***Транзакционным набором*** *называют множество* $D = \{T_1, T_2, \ldots, T_m\}$*, где* T_j *— множество предметов, такое, что* $T_j \subseteq I$*.*

Определение 2. *Множество $T_j \in D$ называется* ***транзакцией****, а число j — её уникальным идентификатором.*

Изначально проблема поиска ассоциативных правил была представлена в терминах «корзины с продуктами». Транзакцию T можно представить как содержимое корзины или чек из супермаркета — множество продуктов, купленных за одно посещение. Транзакционный набор D соответствует базе данных магазина, а множество I — всем доступным продуктам.

Определение 3. *Пусть $X \subseteq I$ — набор предметов. Говорят, что транзакция T_j содержит X, если $X \subseteq T_j$.*

Определение 4. ***Поддержкой*** *(support) набора X называют отношение количества транзакций $T_j \in D$, содержащих X, к мощности D:*

$$sup(X) = \frac{\#\{T_j \in D \mid X \subseteq T_j\}}{|D|}.$$

Определение 5. ***Доверием*** *(confidence) правила $X \Rightarrow Y$ называют условную вероятность:*

$$conf(X \Rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)}.$$

Определение 6. *Правило $X \Rightarrow Y$ называют* ***ассоциативным правилом*** *с параметрами minsup и minconf, если:*

1. $sup(X \cup Y) \geq minsup$,
2. $conf(X \Rightarrow Y) \geq minconf$.

В соответствии с определением выше, будем считать, что ассоциативное правило удерживается на наборе данных, если значение поддержки и доверия правила не меньше минимальных.

3 Описание решения

3.1 Валидация AR

Алгоритм валидации проверяет, удерживается ли заданное ассоциативное правило на конкретных данных. В случае нарушения правила система предоставляет пользователю фактические значения поддержки и доверия, а также транзакции с потенциальными аномалиями.

Процесс валидации строится на расчёте двух коэффициентов для каждой транзакции:

1. Степень соответствия левой части правила (доля элементов левой части в транзакции).

2. Степень соответствия правой части правила.

На основе этих коэффициентов считаются фактические значения поддержки и доверия. Кроме того, алгоритм автоматически группирует транзакции в кластеры ошибок, в которых какая-либо из частей правила представлена не полностью. Такая кластеризация позволяет точечно выявлять источники несоответствий и упрощает анализ данных для их дальнейшей корректировки.

Алгоритм валидации включает в себя несколько этапов:

1. Подсчёт коэффициентов соответствия левой и правой частей правила для каждой транзакции;

2. Вычисление фактических значений поддержки и доверия;

3. Распределение транзакций по ошибочным кластерам.

Для анализа аномалий в данных был разработан специальный тип представления данных — кластеры. В соответствии с описанным выше, каждая транзакция имеет два коэффициента. По этим коэффициентам транзакции распределяются в пять категорий, расположенных в порядке убывания вероятности ошибки:

1. Транзакции, содержащие левую часть правила целиком и частично содержащие правую;

2. Транзакции, содержащие левую часть правила целиком и не содержащие правую;

3. Транзакции, частично содержащие левую часть правила и содержащие правую часть целиком;

4. Транзакции, частично содержащие левую и правую части правила;

5. Транзакции, частично содержащие левую часть правила и не содержащие правую.

Категория и соответствующие ей транзакции формируют кластер. Стоит обратить внимание, что в случаях, когда определённая часть правила состоит из одного элемента, кластера, где эта часть представлена частично, не будет.

3.2 Архитектура

UML-диаграмма архитектуры решения представлена на рисунке 1.

В Desbordante все алгоритмы поиска и валидации наследуются от базового класса `Algorithm`. Он предоставляет интерфейсы для загрузки данных, настройки параметров и выполнения.

Класс `ARVerifier` расширяет эту логику, добавляя методы `GetRealConfidence` и `GetRealSupport` для получения фактических значений доверия и поддержки соответственно. Также `ARVerifier` предоставляет методы для работы с ошибочными транзакциями: их количество и список кластеров.

Для расчёта статистики класс использует агрегированный экземпляр `ARStatsCalculator`. Его метод `CalculateStatistic` вызывается внутри `VerifyAR`, принимая на вход исходные данные и ассоциативное правило. Состояние сбрасывается методом `ResetState` для поддержки повторных запусков.

3.3 Привязки для Python

Для интеграции алгоритма валидации ассоциативных правил в систему Desbordante были разработаны Python-привязки с использованием библиотеки `pybind11`. Этот подход позволяет напрямую связывать C++-реализацию класса `ARVerifier` с Python-интерфейсом, обеспечивая:

- Загрузку транзакционных данных в формате CSV;
- Настройку параметров валидации (минимальная поддержка, доверие);
- Доступ к списку нарушающих транзакций.

Ядро интеграции реализовано в методе `BindARVerification`, где с помощью шаблонных функций создаётся модуль `ar_verification`. Модуль предоставляет Python-класс `ARVerifier` для взаимодействия с алгоритмом валидации.

3.4 Тестирование

Для обеспечения корректности работы алгоритма в различных условиях были написаны 12 модульных тестов, которые проверяют корректность работы алгоритма на различных типах входных данных и при разных значениях поддержки и доверия. Данные для этих тестов были взяты из папки `test_input_data/transactional_data` в репозитории проекта. Тестирование автоматизировано с использованием фреймворка GoogleTest [5].

Для проверки интерфейса валидации ассоциативных правил в Python был разработан тестовый пример[1], демонстрируемый в листинге 1. Он демонстрирует функциональность библиотеки:

1. Загрузка данных: Метод `load_data` загружает табличные данные, указывая формат данных, разделитель и наличие заголовка.

[1]Полная версия примера: https://github.com/Desbordante/desbordante-core/pull/530/files#diff-cfe989b6294ee24652905e2c2b5ab94b4e2ed3ce380ab23cc11e3848ea6f4988

2. Вывод результатов: функция `print_results` выводит, удерживается правило или нет. Если нет, пользователю предоставляются действительные значения поддержки и доверия, а также информация о кластерах.

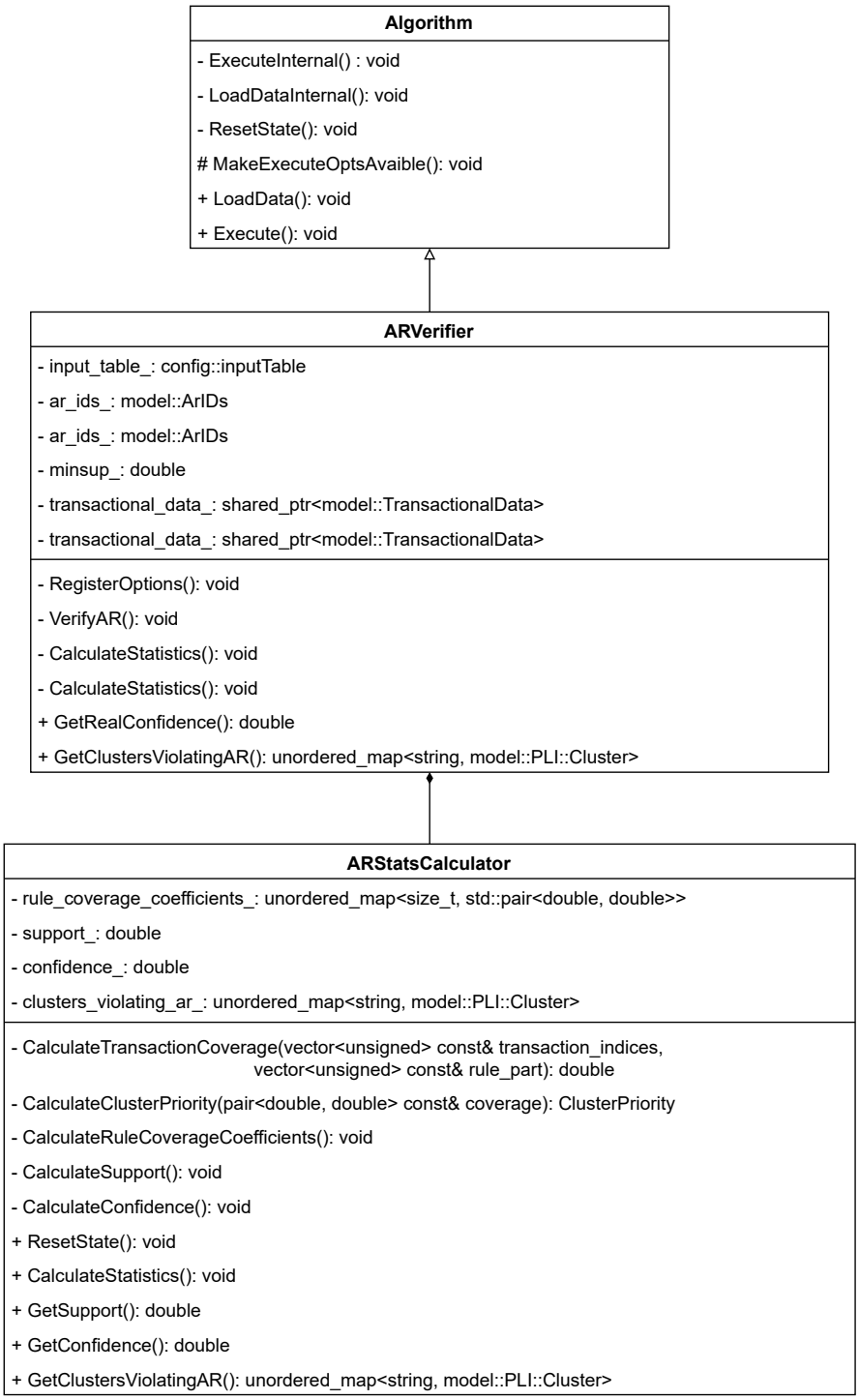

Рис. 1: Диаграмма классов

Листинг 1.: Пример использования в Python

```
import desbordante

TABLE_TABULAR = 'examples/datasets/rules_book_rows.csv'
TABLE_FIXED_TABULAR = ('examples/'
                       'datasets/ar_verification_datasets'
                       '/rules_book_rows_fixed.csv')

def print_results(ar_verifier, supp, conf):
    if ar_verifier.ar_holds():
        print(f'AR holds: {ar_verifier.ar_holds()}')
    else:
        actual_supp, actual_conf = (
            ar_verifier.get_real_support(),
            ar_verifier.get_real_confidence())
        print(f'AR holds: {ar_verifier.ar_holds()}')
        print(f'actual confidence: {actual_conf}')
        print('actual support: {actual_supp}')
        print('Clusters violating AR:')
        clusters_violating_ar =
            ar_verifier.get_clusters_violating_ar()
        for priority, cluster in (
                clusters_violating_ar.items()):
            print(f'{priority} : {cluster}')


algo = desbordante.ar_verification.algorithms.Default()
algo.load_data(table=(TABLE_TABULAR, ',', False),
                   input_format='tabular')
supp, conf = 0.2, 0.6
algo.execute(arule_left=["Bread"], arule_right=["Butter"],
                 minsup=supp, minconf=conf)
print_results(algo, supp, conf)

algo.load_data(table=(TABLE_FIXED_TABULAR, ',', False),
                   input_format='tabular')
algo.execute(arule_left=["Bread"], arule_right=["Butter"],
                 minsup=supp, minconf=conf)
print_results(algo, supp, conf)
```

4 Эксперимент

Основная задача эксперимента — оценить производительность алгоритма на больших объёмах данных.

Были выбраны три набора данных: `groceries.csv`[2] на 9835 транзакций и 32 столбца, `basket.csv`[3] на 14963 транзакций и 11 столбцов,, и `TechIdTransactions2017.csv`[4] на 9835 транзакций и 32 столбца.

Испытания алгоритмов проводились на специализированной тестовой системе со следующими техническими параметрами:

- Операционная система: Arch Linux (Linux Kernel 6.13.4-arch1-1), gcc version – 14.2.1;
- Процессор: 11th Gen Intel(R) Core(TM) i5-1135G7 (8), 4.20 GHz;
- Оперативная память: 16 GiB.

Следует отметить, что из-за отсутствия больших наборов данных на открытых ресурсах, время выполнения проверок не превышало одной секунды, что затрудняет точную оценку. Однако для получения более точных результатов измерения времени проверки проводились путем усреднения по 40 запускам; кроме того, отметим, что значимых выбросов в полученных значениях зафиксировано не было.

Все проверяемые правила были получены с помощью алгоритма Apriori [9]. В итоговое время проверки не входило время на загрузку данных. Нижеприведённые таблицы показывают, что время проверки ассоциативных правил не связано с фактом их выполнения. Это происходит из-за того, что алгоритм подразумевает полное прохождение по всем транзакциям исходного набора данных, прежде чем предъявить результат. Аналогично, время проверки не зависит от значений поддержки и достоверности правил.

[2]groceries https://www.kaggle.com/datasets/irfanasrullah/groceries

[3]basket https://www.kaggle.com/datasets/vikramamin/basket-analysis-association-rule-mining?select=basket.csv

[4]TechId https://www.kaggle.com/datasets/samanemami/online-transactions-of-the-electronic-devices

Таблица 1: Время проверки для `groceries.csv`

Ассоциативное правило	**Время (с)**	**Удерж.**
{*butter, domestic eggs*} → {*whole milk*} (sup=0.003, conf=0.5)	0.260	Да
{*whole milk*} → {*cake bar*} (sup=0.003, conf=0.0015)	0.245	Нет
{*whole milk*} → {*dishes*} (sup=0.2, conf=0.2)	0.254	Нет
{*whole milk*} → {*chewing gum*} (sup=0.001, conf=0.5)	0.256	Нет

Таблица 2: Время проверки для `basket.csv`

Ассоциативное правило	**Время (с)**	**Удерж.**
{*whole milk*} → {*pork*} (sup=0.003, conf=0.0005)	0.207	Да
{*whole milk*} → {*pastry*} (sup=0.003, conf=0.0025)	0.205	Да
{*whole milk*} → {*newspapers*} (sup=0.03, conf=0.0015)	0.205	Да
{*whole milk*} → {*canned beer*} (sup=0.003, conf=0.5)	0.206	Нет

Таблица 3: Время проверки для `TechIdTransactions2017.csv`

Ассоциативное правило	**Время (с)**	**Удерж.**
{*Lenovo Desktop Computer, HP Laptop*} → {*iMac*} (sup=0.015, conf=0.5)	0.321	Да
{*Dell Desktop, Lenovo Desktop Computer*} → {*iMac*} (sup=0.015, conf=0.7)	0.320	Нет
{*Lenovo Desktop Computer, ViewSonic Monitor*} → {*iMac*} (sup=0.015, conf=0.55)	0.324	Да
{*Dell Desktop, ViewSonic Monitor*} → {*HP Laptop*} (sup=0.035, conf=0.56)	0.326	Нет

Заключение

Результатом работы является создание и внедрение функциональности для валидации ассоциативных правил в системе Desbordante. Для достижения этой цели были выполнены следующие задачи:

- Проведён и задокументирован обзор предметной области;
- Разработан алгоритм валидации ассоциативных правил, а также предложена концепция кластеров для анализа исключений;
- Создан набор модульных тестов для проверки на корректность;
- Разработаны Python-привязки для возможности использовать алгоритм верификации в Python, приведён пример использования алгоритма в Python;
- Проведено первичное тестирование производительности.

Исходный код находится на стадии рассмотрения и доступен на GitHub[5].

[5] https://github.com/Desbordante/desbordante-core/pull/530

Список литературы

[1] Agrawal Rakesh, Srikant Ramakrishnan. Fast Algorithms for Mining Association Rules in Large Databases // Proceedings of the 20th International Conference on Very Large Data Bases. — VLDB '94. — San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1994. — P. 487–499. — URL: http://dl.acm.org/citation.cfm?id=645920.672836.

[2] Desbordante: a Framework for Exploring Limits of Dependency Discovery Algorithms / Maxim Strutovskiy, Nikita Bobrov, Kirill Smirnov, George Chernishev // 2021 29th Conference of Open Innovations Association (FRUCT). — 2021. — P. 344–354.

[3] Fan Wenfei, Wu Yinghui, Xu Jingbo. Functional Dependencies for Graphs // Proceedings of the 2016 International Conference on Management of Data. — SIGMOD '16. — New York, NY, USA : Association for Computing Machinery, 2016. — P. 1843–1857. — URL: https://doi.org/10.1145/2882903.2915232.

[4] Frequent pattern mining: Current status and future directions / Jiawei Han, Hong Cheng, Dong Xin, Xifeng Yan // Data Min. Knowl. Discov. — 2007. — 07. — Vol. 15. — P. 55–86.

[5] Google. GoogleTest - Google Testing and Mocking Framework. — https://github.com/google/googletest.

[6] Jakob Wenzel, Rhinelander Jason, Moldovan Dean. pybind11 — Seamless operability between C++11 and Python. — 2017. — URL: https://github.com/pybind/pybind11.

[7] Реализация алгоритма верификации Denial Constraints в Desbordante : Rep. / Saint Petersburg State University ; Executor: Павел Аносов : 2024. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/DC%20Verification%20-%20Anosov%20Pavel%20-%202024%20spring.pdf.

[8] Верификация вероятностных функциональных зависимостей в Desbordante : Rep. / Saint Petersburg State University ; Executor: Никита Немакин : 2024. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/PFD%20Verification%20-%20Nemakin%20Nikita%20-%202024%20spring.pdf.

[9] Реализация функционала для поиска ассоциативных правил в рамках платформы Desbordante : Rep. / Saint Petersburg State University ; Executor: Александр Смирнов : 2022. — URL: https://github.com/Mstrutov/Desbordante/blob/main/docs/papers/Apriori%20-%20Alexandr%20Smirnov%20-%202022%20spring.pdf.