Санкт-Петербургский государственный университет

ВОЛГУШЕВ Иван Романович

Выпускная квалификационная работа

Реализация поиска негативных ассоциативных правил в профилировщике данных Desbordante

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5162.2020 «Технологии программирования»*

Научный руководитель:
доцент кафедры информационно-аналитических систем, к. ф.-м. н. Михайлова Е. Г.

Рецензент:
инженер-разработчик, ООО «Открытая Мобильная Платформа», бакалавр, Фирсов М. А.

Санкт-Петербург
2025

Saint Petersburg State University

Ivan Volgushev

Bachelor's Thesis

Implementing Negative Association Rule Mining in Desbordante Data Profiler

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5162.2020 "Programming Technologies"*

Scientific supervisor:
C.Sc, docent E.G. Mikhailova

Reviewer:
Developer at "Open Mobile Platform" M.A. Firsov

Saint Petersburg
2025

Оглавление

Введение

Профилирование данных — процесс извлечения различных видов метаданных из данных [14]. В профилирование данных входит нахождение простых статистик, таких как количество уникальных или отсутствующих значений в столбце табличной базы данных, определение математического ожидания, дисперсии данных и так далее. Кроме того, существуют и более сложные типы метаданных, описывающие закономерности в данных. Формальные описания таких закономерностей будем называть примитивами. Примитивы могут описывать зависимости между столбцами табличных данных, среди них широко известные функциональные зависимости (functional dependencies), ассоциативные правила (association rules, AR), зависимости включения (inclusion dependencies) и другие.

В этой работе речь пойдёт о поиске негативных ассоциативных правил (negative association rule mining). Классические ассоциативные правила определены на транзакционных наборах данных — наборах, представляющих собой какое-то количество транзакций из фиксированного домена предметов. Каждая транзакция может либо включать предмет из домена, либо нет. Если присутствие предмета A в транзакции с вероятностью P влечёт присутствие предмета B, то говорят, что товары A и B связаны ассоциативным правилом $A \Rightarrow B$ с уровнем доверия (confidence) P. Негативные ассоциативные правила, в свою очередь, являются обобщением классических и допускают не только положительные вхождения предметов в правила, но и отрицательные. То есть, присутствие предмета A в транзакции может говорить об *отсутствии* предмета B. Как и их более узкий аналог, поиск негативных ассоциативных правил применим в решении широкого спектра задач, от рыночного анализа, в контексте которого поиск ассоциативных правил и был предложен [3], до использования для создания классификаторов [11].

Систематический поиск вхождений примитивов зачастую является трудоёмкой задачей, и негативные ассоциативные правила не являются исключением [1]. Для поиска примитивов на больших объёмах

данных требуется оптимизированное программное обеспечение, основанное на развитых алгоритмах поиска. Эту проблему стремится решить наукоёмкий профилировщик данных с открытым исходным кодом Desbordante[1] [7], который предоставляет реализации множества алгоритмов для поиска статистик и вхождений примитивов в табличных и графовых данных. Однако, алгоритмы поиска негативных ассоциативных правил на момент написания данной работы в Desbordante отсутствуют.

С целью расширения функционала Desbordante было решено произвести выбор наиболее подходящего алгоритма поиска негативных ассоциативных правил и реализовать его в проекте. Выбор был сделан в пользу алгоритма Kingfisher [9], работа которого основана на статистических методах.

[1] https://github.com/Desbordante/desbordante-core (дата доступа: 14 мая 2025 г.).

1. Постановка задачи

Целью настоящей работы является интеграция алгоритма Kingfisher для поиска негативных ассоциативных правил в проект Desbordante. Для её выполнения были поставлены следующие задачи:

1. Ознакомиться с предметной областью поиска негативных ассоциативных правил. По результатам ознакомления написать обзор, выбрать алгоритм поиска и обосновать данный выбор;
2. Реализовать алгоритм Kingfisher в рамках проекта Desbordante;
3. Провести эксперименты на реальных данных, изучить производительность полученной реализации.

2. Обзор

Поиск ассоциативных правил является часто используемым инструментом профилирования данных [14]. В этом разделе будет обсуждено, чем может быть полезен поиск *негативных* ассоциативных правил и какие трудности возникают при их поиске в сравнении с их более узким классическим вариантом. Для этого введём основные понятия, связанные с ассоциативными правилами и их поиском.

2.1. Ассоциативные правила: основные определения

Все последующие определения приведены по [3].

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

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

Определение 3. *Пусть* $X \subseteq R$ *— какой-то* ***набор предметов*** *или* ***itemset****. Говорят, что транзакция* T_j *содержит* X*, если* $X \subseteq T_j$

Определение 4. *Пусть* $X \subseteq R$ *— какой-то набор предметов.* ***Поддержкой*** *или* ***support*** *набора* X *называют отношение количества транзакций* $T_j \in D$*, которые содержат* X*, к общему числу транзаций в* D*:*

$$sup(X) = \frac{\{Tj \in D | X \subseteq Tj\}}{|D|}. \tag{1}$$

Определение 5. *Пусть* $X \subseteq R$ *— набор предметов.* X *называется* ***частым набором*** *или* ***frequent itemset****, если его поддержка не меньше некоторой заданной константы* $minsup : sup(X) \geq minsup$*.*

Определение 6. *Пусть* $X, Y \subseteq R$ *— наборы предметов. Доверием или* $confidence$ *правила* $X \Rightarrow Y$ *называют условную вероятность того,*

что набор Y входит в случайно выбранную транзакцию при условии, что набор X также входит в эту транзакцию:

$$conf(X \Rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)}. \quad (2)$$

2.2. Классические алгоритмы поиска

Поддержка и доверие являются популярными способами оценки ассоциативных правил, зачастую задача поиска звучит так: найти все правила, имеющие на входном наборе данных поддержку не менее $minsup$ и доверие не менее $minconf$, где $minsup$ и $minconf$ константы, определяемые пользователем.

Об алгоритмах поиска ассоциативных правил невозможно говорить, не упомянув алгоритмы поиска частых наборов. Большой пласт алгоритмов поиска ассоциативных правил в качестве первого шага находит все частые наборы для какой-то константы $minsup$, выставленной пользователем, а затем создаёт из полученных частых наборов множество всех ассоциативных правил с доверием, не меньшим константы $minconf$, также выбираемой пользователем. При этом поиск частых наборов является значительно более ресурсозатратным шагом, чем генерация ассоциативных правил из них [1]. Таким образом, поиск задача поиска частых наборов выходит на первый план, если необходимо найти ассоциативные правила.

Поиск частых наборов привлёк к себе внимание после основополагающей работы [3], предлагающей поиск ассоциативных правил и частых наборов. С тех пор область стала развитой и широко исследованной. В интернете существует[2] открытый репозиторий, который был создан по итогам FIMI'03 и FIMI'04, научных совещаний, посвящённых реализациям методов поиска частых наборов, которые проводились в рамках конференций IEEE ICDM'03 и IEEE ICDM'04 соответственно. Репозиторий содержит исходный код 21 реализации, среди которых три реализации алгоритма Apriori.

[2] http://fimi.uantwerpen.be/ (дата доступа: 14 мая 2025 г.).

Apriori [4] и подобные (Apriori-like) алгоритмы, к примеру FPTree [8], составляют значительную часть существующих алгоритмов поиска частых наборов. Их работа основывается на генерации частых наборов путём обхода сетки полного перебора наборов, возможных на множестве предметов R. Количество узлов сетки растёт экспоненциально с ростом $d = |R|$, поэтому в реальных условиях обход всех узлов не представляется возможным. Посещается их минимальное подмножество, гарантирующее обнаружение всех наборов с поддержкой как минимум $minsup$, используя **свойство монотонности поддержки**:

Определение 7. *Пусть $X, Y \subseteq I$ — наборы предметов. Тогда для них справедливо свойство, которое называют* ***свойством монотонности поддержки****:*

$$sup(X) \geq sup(Y) \forall X \supseteq Y. \tag{3}$$

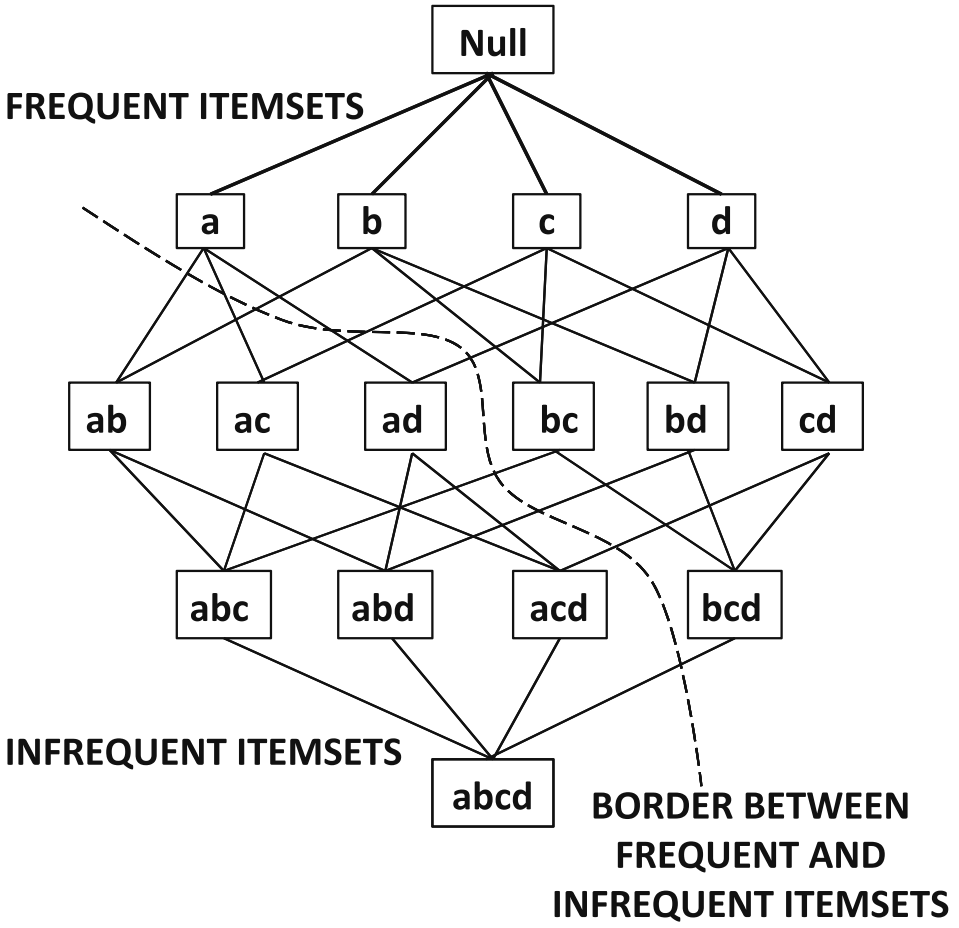

Рис. 1: Сетка обхода кандидатов в частые наборы. Источник: [1]

На Рис. 1 представлена схема сетки кандидатов в частые наборы. В узлах сетки находятся все возможные наборы предметов, на один возможный набор приходится один узел, при этом узлы на одном уровне содержат одинаковое количество предметов. Дерево в зависимости от алгоритма может обходиться как в ширину, так и в глубину, или даже в другом гибридном порядке. При посещении узла производится ресурсозатратная операция — расчёт поддержки соответствующего набора, это может делаться как по определению — полным проходом по всему множеству транзакций и подсчётом тех транзакций, в которые входит набор, так и с применением различных оптимизаций.

В случае, если набор в узле не оказался частым, рассматривать все его надмножества нет необходимости, поскольку, согласно монотонности поддержки, надмножества будут иметь ещё меньшую поддержку и также не будут частыми. При помощи свойства мотонности поддержки в случае обхода в ширину удаётся отбросить большинство узлов сетки и проверить лишь множество наборов, немногим превосходящее искомое.

2.3. Негативные ассоциативные правила: основные определения

Определение 8. ***Негативный предмет*** *определён как* $\neg i_k$. *Если набор* X *содержит негативные предметы, то говорят, что* X *содержится в транзакции* T_j, *если* $i_k \in T_j$ *для всех предметов* i_k *в* X *и* $\neg i_k \notin T_j$ *для всех негативных предметов* $\neg i_k$ *в* X. *Поддержка* $\neg i_k$ *равна* $sup(\neg i_k) = 1 - sup(i_k)$.

Определение 9. ***Негативное ассоциативное правило*** — *это импликация вида* $X \Rightarrow Y$, *где набор* X *и/или* Y *содержит хотя бы один негативный предмет. Негативные ассоциативные правила и наборы не могут содержать одновременно* $\neg i_k$ *и* i_k.

Поиск негативных ассоциаций позволяет извлекать больше закономерностей из данных, чем позволяют обычные правила. В качестве простого примера предположим, что ведётся поиск ассоциативных правил

на базе данных назначений лекарств пациентам. Одна строка — одно назначение от врача. Если какие-то два препарата A и B не встречаются в назначениях вместе, то может существовать правило $A \Rightarrow \neg B$ с поддержкой 0.1 и доверием 0.95. Если изначально о несовместимости препаратов известно не было, можно сделать вывод, что среди врачей распространилось мнение о их несовместимости, и необходимо начать исследование в этом направлении. Поиск классических ассоциативных правил не позволил бы сделать такой вывод, правило $A \Rightarrow B$ имеет поддержку и доверие, близкие к 0.0, поскольку препараты не назначаются вместе, и это правило было бы отброшено.

2.4. Основные проблемы поиска негативных ассоциативных правил

Алгоритмы поиска, рассчитанные на поиск положительных ассоциативных правил, встречаются с множеством проблем при расширении на негативные ассоциативные правила. Эти проблемы приведены ниже:

1. **Гораздо более большое пространство поиска.** На множестве предметов мощностью d существует [3] $3^d - 2^{d+1} + 1$ ассоциативных правил. Это число возрастает [1] до $5^d - 2 \times 3^d + 1$, если учитывать и негативные правила. Число рассматриваемых предметов возрастает в два раза.

2. **Низкая эффективность свойства монотонности поддержки.** Транзакции в транзакционных БД обычно содержат лишь небольшое количество предметов от всего их числа. Пример такой БД приведён в Табл. 6. Зачастую задача поиска ассоциативных правил ставится именно на числе *разреженных* транзакций. Возвращаясь к примеру о базе данных назначений лекарств: очевидно, что каждое назначение будет содержать несколько препаратов, в то время как общее число всех видов препаратов намного больше. Не тяжело представить, что такая БД будет также содержать некоторое число часто назначаемых препаратов и огромное

число препаратов, назначаемых в каких-то редких случаях. Для негативных ассоциативных правил все эти редко назначаемые препараты i_j будут отсечены используя $minconf$ и их надмножества не будут рассмотрены, однако тогда $\neg i_j$ будет обладать достаточной поддержкой.

3. **Большое количество негативных правил неинтересны.** В разряженных транзакционных БД множество негативных правил, ассоциирующих редкие предметы, выйдут на первый план. В Табл. 6 примером такого правила является правило $\neg Q \Rightarrow \neg Z$, имеющее поддержку 0.925 и доверие 1.0.

Таким образом, поиск негативных ассоциативных правил представляет интерес ввиду того, что негативные ассоциативные правила могут нести новую информацию о данных. Однако, их поиск при помощи классических алгоритмов осложнён. В Desborante поиск ассоциативных правил уже представлен алгоритмом Apriori, но его адаптация под поиск негативных правил будет иметь недостатки, описанные выше. Возникает потребность в интеграции нового алгоритма, решающего представленные проблемы.

2.5. Выбор алгоритма

Поиск подходящего для интеграции в Desbordante алгоритма производился среди алгоритмов, описанных в книге [1] в главе, посвящённой поиску негативных ассоциативных правил. Выбор производился, опираясь на следующие требования и предпочтения, составленные в соответствии с нуждами проекта:

- Свойства используемой алгоритмом меры должны быть известны целевой аудитории Desbordante — людям, заинтересованным в профайлинге данных;
- Алгоритм должен быть нацелен на скорость поиска;
- Алгоритм не должен быть стохастическим;

- Алгоритм не должен накладывать большие ограничения на вид правил, среди которых производится поиск. К примеру, поиск негативных ассоциаций только между парами предметов, а не наборов.

- Алгоритм должен иметь значительное количество цитирований на Google Scholar[3].

- Алгоритм должен иметь общий характер и быть применим в широком спектре задач;

Один подход к проблеме поиска негативных ассоциативных правил — выбор других мер их оценки. Также стоит отметить, что все приведённые ниже алгоритмы поиска направлены на обнаружение как классических, так и негативных ассоциативных правил.

- Алгоритм из работы [6], в которой и была впервые предложена задача поиска негативных ассоциативных правил, относительно прост и основан на расчёте специальной меры для наборов.

- Алгоритм, описанный в [2], также относительно прост и использует собственную меру, названную *collective strength*.

- Алгоритмы из [15], [13] фокусируются на исследовании данных о продажах, хотя теоретически применимы и в других сферах. Требуют наличия "таксономии" исследуемых данных.

- Предлагаемый в [10] алгоритм использует собственную меру *mininterest*, авторы не описывают какое значение он должен иметь и что изменяется в результатах когда она меняется. Ещё одним минусом является то, алгоритм ищет правила, связывающие *пары* предметов.

- Алгоритм из [16] — расширение классического алгоритма поиска ассоциативных правил и использует популярную меру *leverage*,

[3] https://scholar.google.com/

однако накладывает жёсткое ограничение на количество предметов в предпосылке и следствии: четыре в предпосылке и один в следствии.

- Алгоритм, описанный в [12], основан на Apriori и разделяет его недостатки. Опирается на принцип открытого дополнения.

- В [5] описан алгоритм, основывающийся на статистической метрике. Также ищет только правила, соединяющие пары предметов.

По итогу исследования выбор пал на алгоритм, основанный на широко применимой статистической мере: точном критерии Фишера, находящий k-лучших правил согласно ей. Это алгоритм Kingfisher из работы [9], он отвечает практически всем поставленным требованиям и предпочтениям. Единственное несоответствие — ограничения, накладываемые им на вид правил, среди которых ведётся поиск: *положительный* набор в предпосылке и негативный или положительный *предмет* в следствии. Быстрый поиск по Google Scholar показал интерес к этому алгоритму. Помимо этого, работа, предложившая Kingfisher, имеет самую недавнюю дату публикации среди прочих упомянутых выше — 2011 год.

3. Описание алгоритма Kingfisher

Алгоритм Kingfisher решает задачу поиска k-лучших по статистической мере p_f классических и негативных ассоциативных правил вида $X \Rightarrow A$, где X это набор положительных предметов, а A это положительный или негативный предмет.

Положительной стороной алгоритма является то, что он может использовать и другие меры оценки правил, авторы алгоритма также предлагают способ использования статистической меры χ^2, однако в этой работе была рассмотрена и реализована только мера p_f.

3.1. Статистическая мера p_f

Мера p_f определена на множестве классических и негативных ассоциативных правил и возвращает вероятность того, что наборы X и Y возникают независимо друг от друга, а наблюдаемая (или более сильная) зависимость появилась случайно. Чем меньше значение меры, тем статистически значимее правило. Формальное определение использует следующую структуру:

Таблица сопряжённости

Для анализа зависимости между предпосылкой X и следствием A используется таблица сопряжённости:

	X	A	Сумма по строкам
Да	a	b	$a+b$
Нет	c	d	$c+d$
Сумма по столбцам	$a+c$	$b+d$	$n=a+b+c+d$

В данной таблице a, b, c, d — наблюдаемые частоты совместных событий, а n — общее число наблюдений.

Нулевая гипотеза предполагает независимость атрибутов. Для её проверки рассматриваются все возможные таблицы с теми же маргинальными суммами (т.е. с сохранением сумм по строкам и столбцам

исходной таблицы).

Формула для меры

Мера p_f вычисляется как сумма вероятностей всех таблиц, которые демонстрируют более сильное отклонение от нулевой гипотезы, чем наблюдаемая таблица:

$$p_f(X \Rightarrow A = a) = \sum_i \frac{\binom{a_i + b_i}{a_i}\binom{c_i + d_i}{c_i}}{\binom{n}{a_i + c_i}},$$

где: $\binom{a_i + b_i}{a_i}$ — число способов выбрать a_i "успехов" в группе X при фиксированной сумме строки, $\binom{c_i + d_i}{c_i}$ — аналогично для группы A, $\binom{n}{a_i + c_i}$ — общее число способов распределить наблюдения при фиксированных маргинальных суммах.

3.2. Обход дерева кандидатов.

Устройство дерева

Kingfisher в ходе работы обходит в ширину дерево кандидатов (Рис. 2) — структуру, каждый узел которой соответствует некоторому набору Z. Каждому возможному набору соответствует единственный узел, атрибуты упорядочены по частоте их появления в базе данных. Обход такого дерева в ширину эквивалентен лексикографическому перебору сочетаний атрибутов. На Рис. 2 закрашен узел ACD и его *прародители*: AC, AD и CD — узлы, соответствующие подмножествам набора ACD.

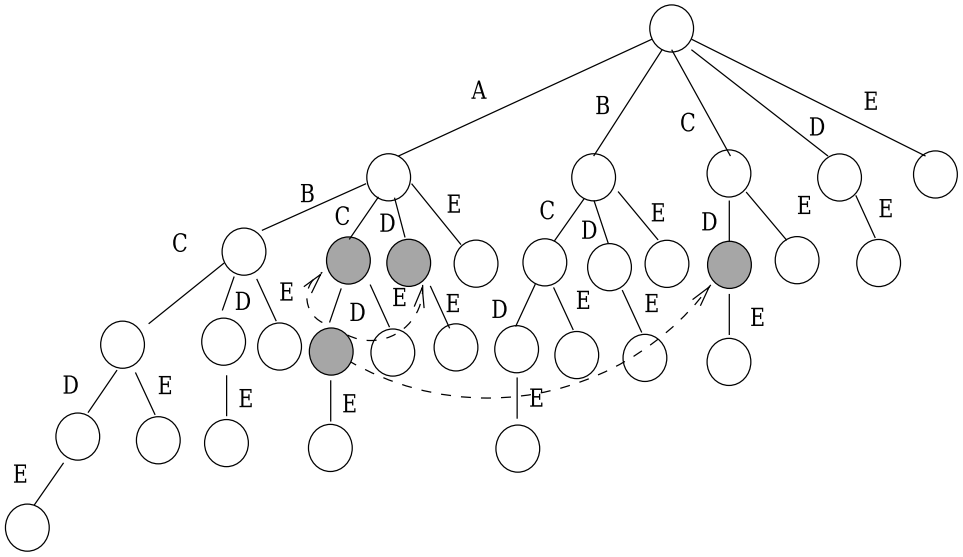

Рис. 2: Устройство префиксного дерева обхода правил. Темным цветом выделен узел ACD и его прародители. Источник: [9]

При посещении узла Z рассматриваются все правила вида $X \Rightarrow A = a$, где $X = Z \setminus A$, $a \in \{0, 1\}$. Для каждого правила рассчитывается p_f. Посетив все узлы, будут оценены все возможные на R правила вида $X \Rightarrow A = a$ и поставленная задача поиска k-лучших по p_f будет решена.

Исключение веток дерева

Посещение всех узлов, как и в случае с Apriori-like сеткой кандидатов, не представляется возможным. В каждом узле Z дерева хранится два массива бит длины d: *ppossible* и *npossible* (Рис. 3) Если бит номер k в массиве *ppossible* равен нулю, то правило со следствием $A_k = 1$ не попадёт в k-лучших в Z и его потомках. Биты в *npossible* несут ту же информацию, но для следствия $A_k = 0$.

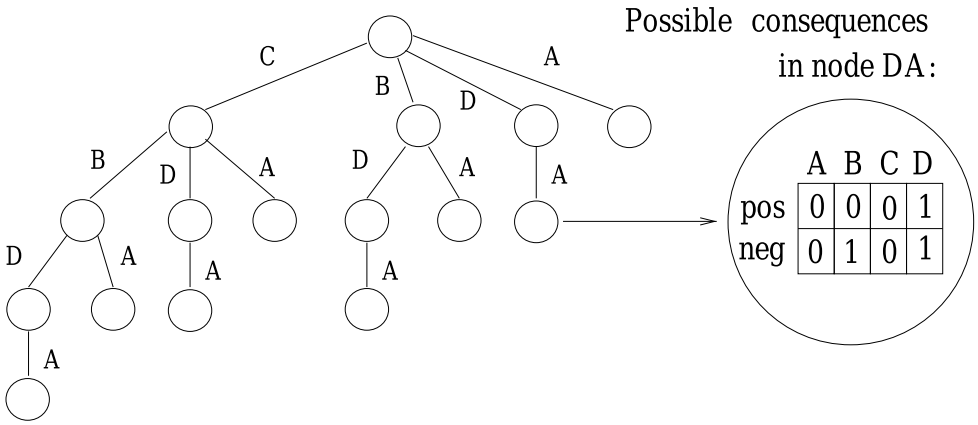

Рис. 3: Битовые массивы *npossible* и *ppossible* в вершине DA. Возможные следствия для этой вершины и её потомков это D, $\neg B$ и $\neg D$. Источник: [9]

Таким образом, каждому возможному правилу сопоставляется бит в дереве, если бит равен нулю, то соответствующее правило не оценивается в узле и его потомках. Kingfisher использует три механизма для выставления нулевых значений битов, используя три оценки нижних границ p_f, которые были предложены и доказаны авторами:

Нижние границы для p_f:

$$lb_1 = \frac{m(A)!\, m(\neg A)!}{n!}, \tag{4}$$

$$lb_2 = \frac{m(\neg X)!\, m(A = a)!}{n!\, (m(A = a) - m(X))!}, \tag{5}$$

$$lb_3 = \frac{m(A)!\, m(\neg A)!\, (n - m(XA = a))!}{n!\; m(\neg X\, A = a)!\; m(A \neq a)!}. \tag{6}$$

где $m(X)$ — количество транзакций включающих X и n — количество транзакций в D

1. **Наследование нулевых бит.** Все потомки узла, в котором для $A = a$ бит обнулён, автоматически наследуют этот нулевой бит и не рассматривают соответствующее правило.

2. **Правило максимального p.** Во время поиска хранится и обновляется максимальное (худшее) значение p_f среди k-лучших обна-

руженных правил. Если оценка значения p_f для правила оказалась больше этого максимума, соответсвующий бит обнуляется.

3. **Правило избыточности.** В каждом узле хранится наименьшее значение p_f среди всех правил предков Z. Правила признаются избыточными и соответствующий бит обнуляется в узле Z, если их p_f не лучше (не меньше), чем минимальное p_f среди более общих правил $W \subset Z$.

4. **Принцип *Lapis Philosophorum*.** Если в узле Z для данного $A_k = a$ соответствующий бит равен нулю и $A_k \in Z$, то он может быть обнулён и в $Z \setminus A_k$.

Таким образом, при посещении узла Z алгоритм совершает следующие шаги:

- При посещении узла Z формируются массивы *ppossible* и *nposible* путём обнуления бит, которые обнулены хотя бы в одном родителе.

- Для каждого ещё не отброшенного правила рассчитывается нижняя граница p_f, и если в следствии *правила максимального p* или *правила избыточности* бит был отброшен, правило игнорируется и соответствующий бит в Z обнуляется.

- В случае обнуления оно может переностись дальше по дереву при помощи *Lapis Philosophorum* и соответствующий бит обнулится и в родителях.

- Иначе правило оценивается, и если оно оказалось достаточно хорошим, обновляются глобальное максимальное p_f и максимальное p_f в Z, после чего правило добавляется в k-лучших.

- Алгоритм переходит к следующему узлу.

Таким образом, алгоритм Kingfisher эффективно решает задачу поиска k-лучших ассоциативных правил по мере p_f за счёт комбинации

обхода дерева кандидатов в ширину и раннего отсечения ветвей. Использование трёх нижних границ p_f, наследования нулевых бит, правила максимального и избыточности позволяет существенно сократить число оцениваемых правил. Принцип *Lapis Philosophorum* дополнительно ускоряет поиск, передавая обнуления родителям. В результате алгоритм обеспечивает получение статистически значимых и неизбыточных правил с высокой производительностью.

3.3. Описание реализации

В качестве базового класса, ответственного за определение основных возможных опций, общих для всех алгоритмов поиска негативных ассоциативных правил был создан класс `NeARDiscovery`, унаследованный от базового класса Desbordante `Algorithm`. Система опций Desbordante позволяет классам, наследующим от `Algorithm` передавать в систему опций адреса своих полей, значения которым могут затем быть безопасно присвоены пользователем через CLI или Python-интерфейс.

Класс `Kingfisher` же в свою очередь унаследован от `NeARDiscovery` и берёт на себя определение опций, специфичных для алгоритма Kingfisher. При вызове основной функции `Execute` класс `Kingfisher` производит предварительные рассчёты частоты атрибутов и их сортировку а затем создаёт класс дерева поиска `CandidatePrefixTree`. UML-диаграмма этого класса представлена на Рис. 6.

Когда пользователь использует Kingfisher, его создание происходит при помощи фабричного метода `CreateAndLoadAlgorithm(StdParamsMap const& options)`.

Класс CandidatePrefixTree реализует основную структуру алгоритма — префиксное дерево для обхода кандидатов в k-лучших ассоциативных правил (NeAR).

Для расчёта точного критерия Фишера для меры p_f был использован объект `boost::math::hypergeometric_distribution` из широко используемой библиотеки Boost, позволяющий рассчитать приближённое значение меры. Boost уже используемая проектом библиотека, что

Листинг 1: Вывод новой реализации.

```
3.947e-18  {3, 0} -> not 1
5.777e-14  {3} -> not 2
5.777e-14  {1, 0} -> 2
5.777e-14  {1, 0} -> not 3
1.735e-10  {1} -> 2
1.735e-10  {3} -> not 1
1.735e-10  {1} -> not 3
1.735e-10  {2} -> 1
```

позволило избежать добавления новых зависимостей. Функция расчёта метрики p_f представлена в листинге ниже.

3.4. Тестирование

Для подтверждения корректности новой реализации Kingfisher был создан тестовый сценарий, повторяющий пример из оригинальной статьи:

Таблица 1: Тестовый сценарий из оригинальной статьи.

set	**freq.**
$ABC\neg D$	10
$A\neg B\neg CD$	85
$\neg AB\neg CD$	5

Таблица 2: Правила, обнаруженные реализацией из оригинальной статьи.

Rule	p_F
$AD \Rightarrow \neg B$	$3.9 \cdot 10^{-18}$
$D \Rightarrow \neg C$	$5.8 \cdot 10^{-14}$
$AB \Rightarrow C$	$5.8 \cdot 10^{-14}$
$AB \Rightarrow \neg D$	$5.8 \cdot 10^{-14}$
$C \Rightarrow B$	$1.7 \cdot 10^{-10}$
$D \Rightarrow \neg B$	$1.7 \cdot 10^{-10}$

```
double GetFishersP(model::NeARIDs const& rule,
                   std::shared_ptr<model::TransactionalData>
                   transactional_data) {
    // Лямбда-функции для проверки условий
    auto hasAnte = //. . .
    auto hasCons = //. . .
    // Подсчет элементов таблицы 2x2
    std::size_t a = 0, b = 0, c = 0, d = 0;
    for (auto const& [_, tx] :
    transactional_data->GetTransactions()) {
        bool X = hasAnte(tx.GetItemsIDs(), rule);
        bool Y = hasCons(tx.GetItemsIDs(), rule);

        if (X && Y)      ++a;
        else if (X)      ++b;
        else if (Y)      ++c;
        else             ++d;
    }
    // Параметры гипергеометрического распределения
    unsigned const N = a + b + c + d;
    unsigned const r = a + c;
    unsigned const n = a + b;

    using namespace boost::math;
    hypergeometric_distribution<> dist(r, n, N);
    // Расчет границ распределения
    int const support_min = std::max(0, int(n + r - N));
    int const quantile = int(a) - 1;

    if (quantile < support_min) {
        return 1.0;  // Крайний случай
    }
    // Вычисление p
    return cdf(
        complement(dist, static_cast<double>(quantile))
    );
}
```

Рис. 4: Функция расчёта точного критерия Фишера.

В Листинге 1 представлен вывод новой реализации, она использует индексы атрибутов, а не их названия, и два значимых правила из оригинального вывода продублированы в симметричном виде. Это происходит ввиду того, что мера p_f симметрична для случая двух атрибутов и оригинальная реализация выводит версию симметричных правил, имеющую наибольшую поддержку. В остальном выводы идентичны.

Таким образом была протестирована корректность новой реализации.

4. Эксперимент

Авторы алгоритма Kingfisher в работе [9] называют принцип отсечения ветвей *Lapis Philosophorum* ключевым способом оптимизации алгоритма. Было решено провести эксперимент с целью определения того, как *Lapis Philosophorum* влияет на производительность созданной для Desbordante реализации.

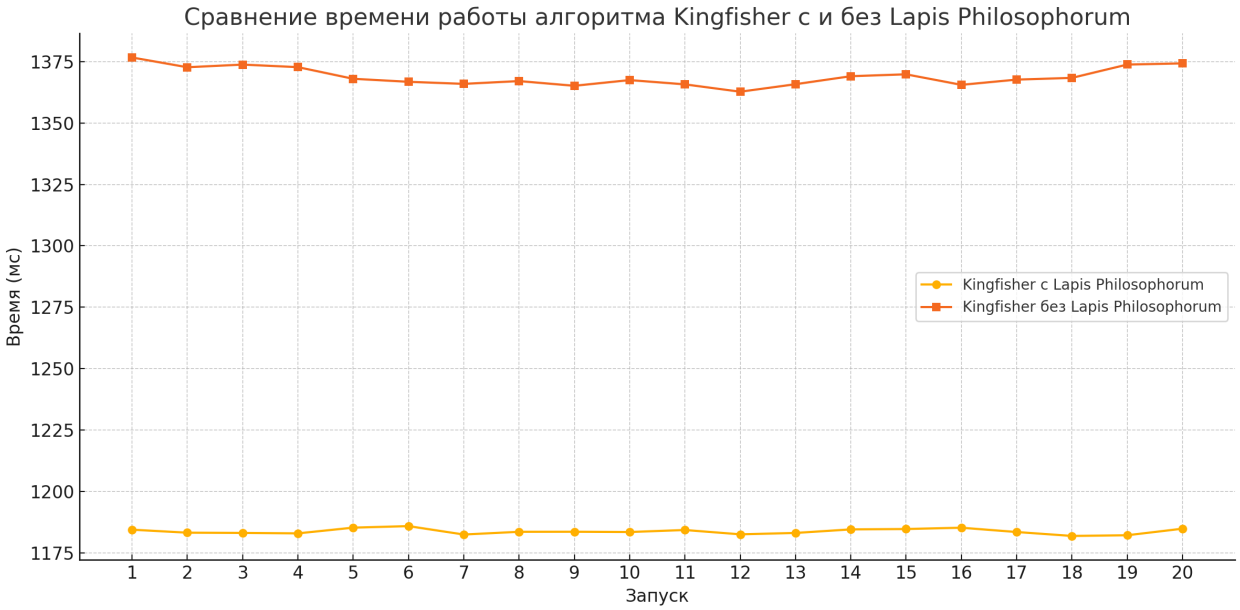

Рис. 5: Сравнение времени работы Kingfisher без принципа *Lapis Philosophorum* и с ним.

Параметры эксперимента

В обоих случаях алгоритм Kingfisher исследовал дерево до уровня 3 включительно. Во втором случае поиск не завершался целиком из-за достижения лимита памяти в 9 ГБ.

Информация о датасете *Chess*

- Число рядов: 3196
- Число атрибутов: 75
- Средняя длина транзакции: 37.0

Выводы

Средние значения времени исполнения алгоритма:

- С применением принципа *Lapis Philosophorum*: **1183.68**
- Без применения принципа *Lapis Philosophorum*: **1368.36**

Без применения принципа *Lapis Philosophorum алгоритм* завершал работу досрочно ввиду нехватки памяти.

Таким образом, принцип *Lapis Philosophorum* демонстрирует значительное улучшение производительности.

5. Заключение

В ходе работы была произведена интеграция алгоритма Kingfisher для поиска негативных ассоциативных правил в проект Desbordante. Были выполнены следующие задачи:

1. Произведено ознакомление с областью поиска негативных ассоциативных правил и по результатам ознакомления был написан написать обзор, выбран алгоритм поиска Kingfisher и этот выбор был обоснован;
2. Реализован алгоритм Kingfisher в рамках проекта Desbordante;
3. Проведены эксперименты на реальных данных, изучена производительность полученной реализации.

Реализация доступна на GitHub[4].

[4] https://github.com/VanyaVolgushev/desbordante-core/tree/add-kingfisher (дата доступа: 14 мая 2025 г.).

Список литературы

[1] Aggarwal Charu C., Bhuiyan Mansurul A., Hasan Mohammad Al. Frequent Pattern Mining Algorithms: A Survey // Frequent Pattern Mining / Ed. by Charu C. Aggarwal, Jiawei Han. — Cham : Springer International Publishing, 2014. — ISBN: 978-3-319-07821-2. — URL: https://doi.org/10.1007/978-3-319-07821-2_2.

[2] Aggarwal Charu C., Yu Philip S. A new framework for itemset generation // Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. — PODS '98. — New York, NY, USA : Association for Computing Machinery, 1998. — P. 18–24. — URL: https://doi.org/10.1145/275487.275490.

[3] Agrawal Rakesh, Imieliński Tomasz, Swami Arun. Mining association rules between sets of items in large databases // Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. — SIGMOD '93. — New York, NY, USA : Association for Computing Machinery, 1993. — P. 207–216. — URL: https://doi.org/10.1145/170035.170072.

[4] 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.

[5] Antonie Luiza, Zaïane Osmar. Mining Positive and Negative Association Rules: An Approach for Confined Rules // European Conference on Principles of Data Mining and Knowledge Discovery. — Vol. 3202. — 2004. — 11. — P. 27–38.

[6] Brin Sergey, Motwani Rajeev, Silverstein Craig. Beyond market baskets: generalizing association rules to correlations // Proceedings of the 1997 ACM SIGMOD International Conference on Management of

Data. — SIGMOD '97. — New York, NY, USA : Association for Computing Machinery, 1997. — P. 265–276. — URL: https://doi.org/10.1145/253260.253327.

[7] 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.

[8] Han Jiawei, Pei Jian, Yin Yiwen. Mining frequent patterns without candidate generation // Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. — SIGMOD '00. — New York, NY, USA : Association for Computing Machinery, 2000. — P. 1–12. — URL: https://doi.org/10.1145/342009.335372.

[9] Hämäläinen Wilhelmiina. Kingfisher: An efficient algorithm for searching for both positive and negative dependency rules with statistical significance measures // Knowledge and Information Systems - KAIS. — 2011. — 08. — Vol. 32. — P. 1–32.

[10] Kishor Peddi, Porika Sammulal. An efficient approach for mining positive and negative association rules from large transactional databases // 2016 International Conference on Inventive Computation Technologies (ICICT). — Vol. 1. — 2016. — P. 1–5.

[11] Liu Bing, Hsu Wynne, Ma Yiming. Integrating Classification and Association Rule Mining // Proceedings of 4th International Conference on Knowledge Discovery Data Mining (KDD). — 1970. — 02.

[12] Mining Positive and Negative Association Rules from Large Databases / Chris Cornelis, Peng Yan, Xing Zhang, Guoqing Chen // 2006 IEEE Conference on Cybernetics and Intelligent Systems. — 2006. — P. 1–6.

[13] Mining negative association rules / Xiaohui Yuan, B.P. Buckles, Zhaoshan Yuan, Jian Zhang // Proceedings ISCC 2002 Seventh Inter-

national Symposium on Computers and Communications. — 2002. — P. 623–628.

[14] Naumann Felix. Data profiling revisited // SIGMOD Rec. — 2014. — . — Vol. 42, no. 4. — P. 40–49. — URL: https://doi.org/10.1145/2590989.2590995.

[15] Savasere A., Omiecinski E., Navathe S. Mining for strong negative associations in a large database of customer transactions // Proceedings 14th International Conference on Data Engineering. — 1998. — P. 494–502.

[16] Thiruvady Dhananjay R., Webb Geoff I. Mining Negative Rules Using GRD // Advances in Knowledge Discovery and Data Mining / Ed. by Honghua Dai, Ramakrishnan Srikant, Chengqi Zhang. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2004. — P. 161–165.

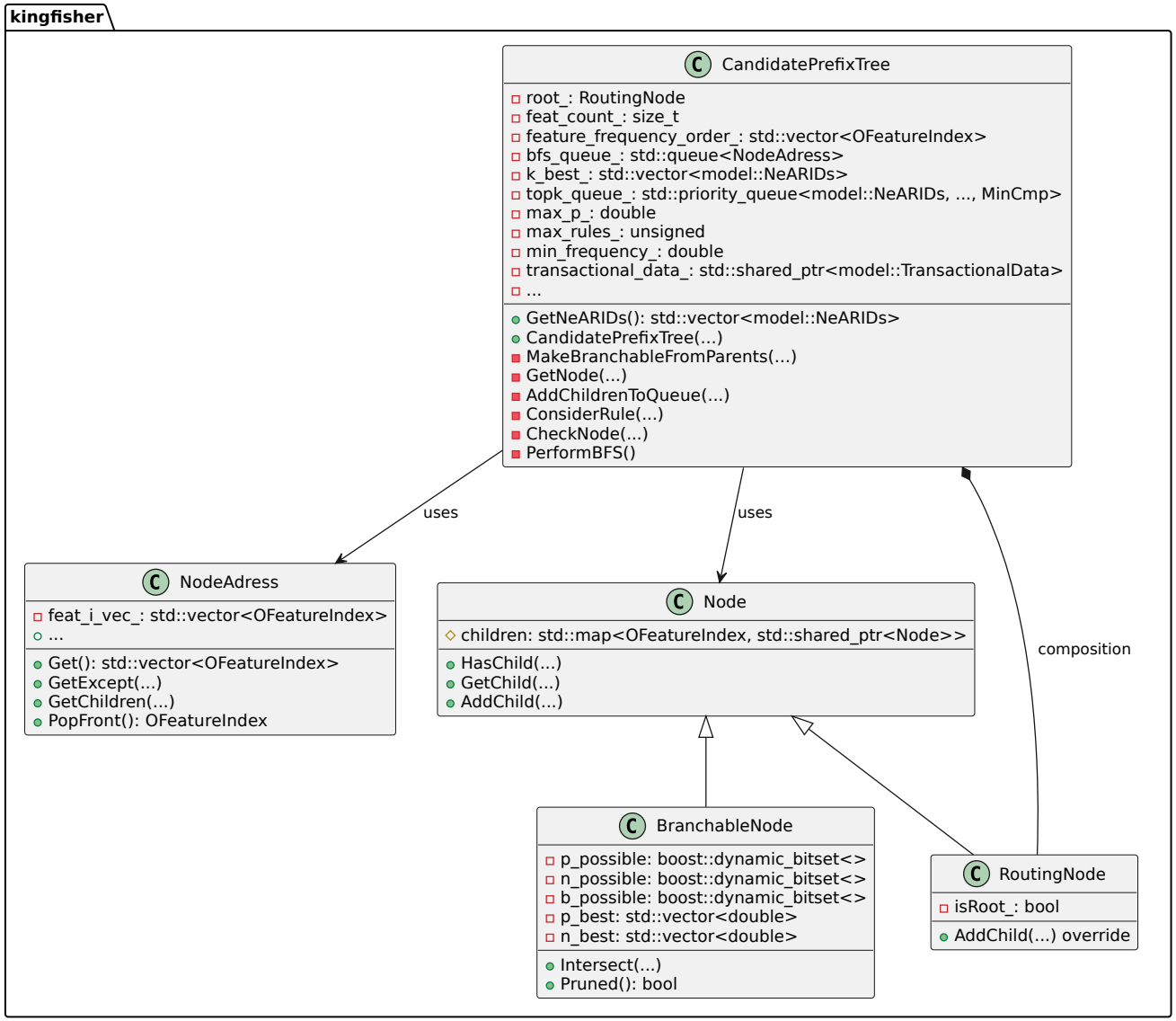

Рис. 6: UML-диаграмма класса `CandidatePrefixTree`.

Таблица 3: Пример разреженной транзакционной базы данных.