Санкт-Петербургский государственный университет

Кафедра иноформационно-аналитических систем

Группа 22.Б09-мм

Реализация алгоритма CORDS для поиска «мягких» функциональных зависимостей и корреляций

Шалашнов Егор Дмитриевич

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
Ассистент кафедры ИАС Чернышев Г. А.

Санкт-Петербург
2024

Оглавление

Введение

Профилирование данных (data profiling), представляет собой процесс извлечения метаданных (т.е. данных о данных) из набора данных. Профилирование подразделяют на классическое и наукоёмкое. Классическое выделяет такие метаданные как: размер файла, время создания, авторство и т.д. Наукоемкое же занимается выявлением различных закономерностей внутри данных [7]. Такие закономерности могут быть важны для различных областей научной деятельности.

Desbordante — высокопроизводительная платформа для профилирования данных с открытым исходным кодом[1], разработанная с использованием языка программирования C++. Данная платформа способна обнаруживать и подтверждать различные закономерности, используя разнообразные алгоритмы. Desbordante поддерживает такие типы закономерностей как: функциональные зависимости, условные функциональные зависимости, метрические функциональные зависимости и многое другое [4].

Важной характеристикой баз данных являются функциональные зависимости (ФЗ). Они задают строгое соответствие между атрибутами фиксированного отношения в терминах реляционной модели. Поиск ФЗ, удерживающихся над отношением, представляет важную задачу в анализе данных. Одним из возможных обобщений ФЗ являются мягкие функциональные зависимости (МФЗ). В отличии от функциональных зависимостей, наличие МФЗ означает, что одни значения определяют другие, не точно, а лишь с высокой вероятностью [6].

Алгоритм CORDS [1] позволяет автоматически выявлять МФЗ и корреляции в табличных данных. Данная работа представляет собой отчёт о реализации предварительной версии алгоритма CORDS, для его последующей интеграции в платформу Desbordante.

[1] https://github.com/Mstrutov/Desbordante/

1. Постановка задачи

Целью работы является реализация алгоритма CORDS для поиска «мягких» функциональных зависимостей и корреляций. Для достижения цели были поставлены следующие задачи:

1. провести обзор алгоритма CORDS;
2. реализовать алгоритм на языке программирования C++;
3. произвести тестирование алгоритма;

2. Обзор алгоритма CORDS

Несмотря на то что в реляционной модели данных термины атрибут и столбец, а также отношение и таблица не являются взаимозаменяемыми, в данной работе мы употребляем их в качестве синонимов, хотя и понимаем различия.

2.1. Предпосылки

В отличии от жёстких ограничений (таких как ФЗ), мягкие функциональные зависимости могут быть полезны для улучшения оценки селективности при оптимизации запросов [1], путём сбора совместной статистики для коррелирующих столбцов. Кластеризованный индекс по таким атрибутам может быть использован для оптимизации запроса.

МФЗ также могут быть использованы для повышения производительности обработки запросов, благодаря тому свойству, что при наличии МФЗ значения одного атрибута хорошо предсказываются значениями некоторых других. Если столбец коррелирует с другим столбцом, можно рекомендовать вторичные индексы [2].

МФЗ и корреляции представляют интерес и сами по себе, так как могут быть использованы для извлечения данных (data mining), например, на их основе может быть построен граф зависимостей таблицы.

Алгоритм CORDS позволяет найти мягкие функциональные зависимости и корреляции в таблице данных.

2.2. Основные понятия

Алгоритм носит название CORDS (CORrelation Detection via Sampling), так как основная идея заключается в том, что для выявления зависимостей используется не вся исходная таблица, а её случайная выборка.

Большинство определений ниже взято из [1].

Определение 1. *Функциональная зависимость (ФЗ) имеет вид*

$$FD : X \to Y,$$

где X, Y это атрибуты отношения R, и означает, что для любых двух кортежей из экземпляра отношения R, если они имеют одинаковые значения X, то и их значения Y должны быть одинаковыми [6].

Определение 2. *Мягкая функциональная зависимость (МФЗ) имеет вид*

$$SFD : X \Rightarrow_s Y,$$

где X, Y — атрибуты некоторого отношения R, а s — минимальный порог меры силы МФЗ (англ. threshold of strength measure). Мерой силы называют

$$S(X \Rightarrow Y, r) = \frac{|X|_r}{|XY|_r},$$

где $|X|_r$ — количество различных значений в столбце X в r, а $|XY|_r$ — это количество различных значений в конкатенации X и Y в r. Мера силы определяет, на каком уровне $X \Rightarrow Y$ имеет место в экземпляре отношения r. МФЗ удерживается, если $S(X \Rightarrow Y, r) \geq s$. Таким образом, при наличии МФЗ, значение X определяет значение Y не с точностью, а с высокой вероятностью [6].

Таблица 1: Пример экземпляра отношения r с МФЗ

	Name	Address	Region	Rate
t_1	Hyatt	175 North Jackson Street	Jackson	230
t_2	Hyatt	175 North Jackson Street	Jackson	250
t_3	Hyatt	6030 Gateway Boulevard E	El Paso	189
t_4	Hyatt	6030 Gateway Boulevard E	El Paso, TX	189

В данном примере[2]:

$$S(\text{address} \Rightarrow \text{region}, r) = \frac{2}{3},$$

[2]Пример взят из [6].

$$S(\text{name} \Rightarrow \text{address}, r) = \frac{1}{2},$$

что означает, что address $\Rightarrow$ region почти удерживается, в то время как МФЗ name $\Rightarrow$ address скорее не удерживается, при $s \leq \frac{1}{2}$.

Замечание. *В нашей терминологии функциональная зависимость является частным случаем МФЗ, с мерой силы равной единице.*

Определение 3. *Мягким ключом параметра* ϵ, $\epsilon \in (0, 1)$ *назовем атрибут* C, *такой что* $|C| >= (1 - \epsilon)|R|$, *где* $|C|$ — *число различных значений столбца,* $|R|$ — *число строк в исходной таблице* R.

Определение 4. *Тривиальным атрибутом (столбцом) назовем атрибут* C, *такой что* $|C| = 1$.

Определение 5. *Кандидатом назовём тройку* (a_1, a_2, P), *где* a_i $(i = 1, 2)$ *это атрибут (столбец) таблицы,* P — *правило сопряжения, указывающее какие конкретно значения* a_1 *сопоставляются каким конкретно значениями* a_2. *Случай, когда столбцы* a_1 *и* a_2 *лежат в одной таблице, называется тривиальным правилом сопряжения.*

Зачастую, P — тривиальное правило или JOIN предикат, поэтому в приведенной реализации рассматриваются только кандидаты с тривиальным правилом сопряжения.

Определение 6. *Рассмотрим таблицу* R *состоящую из 2-х столбцов* A *и* B, *разобьём их значения на* J *и* K *категорий соответственно. Назовём таблицей сопряженности таблицу вида:*

	B_1	B_2	$\dots$	$B_k \dots$	B_K	Row Totals
A_1	f_{11}	f_{12}	$\dots$	$f_{1k} \dots$	f_{1K}	f_{1+}
A_2	f_{21}	f_{22}	$\dots$	$f_{2k} \dots$	f_{2K}	f_{2+}
A_j	f_{j1}	f_{j2}	$\dots$	$f_{jk} \dots$	f_{jK}	f_{j+}
A_J	f_{J1}	f_{J2}	$\dots$	$f_{Jk} \dots$	f_{JK}	f_{J+}
Column totals	f_{+1}	f_{+2}	$\dots$	$f_{+k} \dots$	f_{+K}	$f_{++} = N$

Здесь f_{jk} — *частота появления в таблице* R *пар значений (категорий) вида* (a_t, b_t), *где* $a_t \in A_j, b_t \in B_k$; f_{+k}, f_{j+} — *суммы частот по*

столбцам и строкам соответственно; N — общее число пар значений (категорий), представленных в таблице R [5].

Определение 7. *f_{jk} равный нулю назовём структурным нулём.*

2.3. Параметры алгоритма

- $minimum\_cardinality$ — параметр мягкого ключа;
- $max\_diff\_vals\_proportion$ — максимальная доля, которую должно составлять количество различных значений в конкатенации столбцов выборки от количества строк в выборке, чтобы проверка на наличие мягкой функциональной зависимости имела смысл;
- $min\_sfd\_strength\_measure$ — минимальный порог меры силы мягкой функциональной зависимости;
- $(1 - min\_skew\_threshold)|R|$ — минимальная сумма частот появления самых частых значений столбца в таблице R, указывающая на то, что распределение значений в столбце перекошено;
- $min\_structural\_zeroes\_amount\_d_1 \times d_2$ — минимальное количество структурных нулей в таблице сопряженности, указывающее на то, что столбцы коррелируют. d_1, d_2 — количество различных значений в столбцах C_1, C_2 соответственно;
- $max\_false\_positive\_probability$ — максимальная допустимая вероятность ложно-положительного результата теста на наличие корреляции;
- δ — вспомогательная константа для вычисления размера выборки, $\delta > max\_diff\_vals\_proportion$;
- $max\_amount\_of\_categories$ — максимальное количество категорий для теста на наличие корреляции;

При этом, все параметры лежат в интервале $(0, 1)$.

2.4. Описание алгоритма

Algorithm: DetectCorrelation
INPUT : A column pair C_1, C_2 **with** $|C_1|_R \geq |C_2|_R$

Discover Trivial Cases
1 IF $|C_i|_R \geq (1-\epsilon_1)|R|$ for i = 1 or i = 2
THEN C_i is a soft key; RETURN.
IF $|C_i|_R = 1$ for i = 1 or i = 2
THEN C_i is a trivial column; RETURN.

Sampling
2 Sample R to produce a reduced table S.

Detect Soft Functional Dependencies in the Sample
3 Query S to get $|C_1|_S, |C_2|_S$ and $|C_1, C_2|_S$
IF $|C_1, C_2|_S \leq \epsilon 2|S|$
AND $|C_1|_S \geq (1-\epsilon_3)|C_1, C_2|_S$
THEN $C_1 \Rightarrow C_2$; RETURN.

Skew Handling for Chi-Squared Test
4 FOR $i = 1, 2$
IF $\sum_{j=1}^{N_i} F_{ij} \geq (1-\epsilon_4)|R|$
THEN
SKEW$_i$ = TRUE;
$d_i = N_i$;
FILTER = "C_i IN $\{V_{i1}, \ldots, V_{iN_i}\}$"
ELSE
SKEW$_i$ = FALSE;
$d_i = min(|C_i|_R, d_{max})$;
FILTER = NULL.
Apply FILTER.

Sampling-Based Chi-Squared Test
5 Initialize each $n_{ij}, n_{i\cdot}$ and $n_{\cdot j}$ to 0.
FOR EACH column-value pair(x_1, x_2)
$i = Category(1, x_1, d_1$,SKEW$_1)$;
$j = Category(2, x_2, d_2$,SKEW$_2)$;
Increment $n_{ij}, n_{i\cdot}$ and $n_{\cdot j}$ by 1;
IF $\sum_{i=1}^{d_1} \sum_{j=1}^{d_2} IsZero(n_{ij}) > \epsilon_5 d_1 d_2$
THEN C_1 and C_2 are correlated; RETURN.
Compute χ^2; set $\nu = (d_1 - 1)(d_2 - 1)$
and $t = G_\nu^{-1}(1-p)$.
IF $\chi^2 > t$
THEN C_1 and C_2 are correlated; RETURN.
ELSE C_1 and C_2 are independent; RETURN.

Рис. 1: Алгоритм выявления МФЗ и корреляция в паре столбцов. Листинг взят из статьи [1].

Алгоритм для поиска МФЗ и корреляций представлен на рисунке 1. Непосредственно перед запуском алгоритма, для каждого столбца поступившей таблицы необходимо вычислить число различных значений в нём и число появлений в столбце для каждого значения.

Шаг 1, на вход алгоритму поступает кандидат. Если один из столбцов кандидата тривиален или является мягким ключом, то корреляция считается тривиальной и алгоритм завершает работу.

Шаг 2, алгоритм строит по кандидату выборку с заменой S. Размер выборки вычисляется по формуле:

$$n \approx \frac{[-16\nu \log(p\sqrt{2\pi})]^{1/2} - 8\log(p\sqrt{2\pi})}{1.69\delta(d-1)\nu^{-0.071}}, \quad (1)$$

где d_1, d_2 — число различных значений в столбцах C_1, C_2 соответственно, $d = min(d_1, d_2)$, $\nu = (d_1 - 1)(d_2 - 1)$.

Шаг 3, построенная выборка проверяется на наличие мягкой функциональной зависимости. При наличии МФЗ алгоритм прекращает работу.

Шаг 4, если МФЗ не была обнаружена, проверяется, не перекошено ли распределение значений в столбцах исходной таблицы. Для этого, у каждого столбца вычисляется сумма частот появления самых частых значений $\Sigma_{j=1}^{N_i} F_{ij}$ и сравнивается с величиной $(1 - \epsilon_4)|R|$. Здесь N_i — число самых часто встречающихся значений, $N_i = min(d_{max}, |C_i|)$, F_{ij} — частота появления j-го самого частого значения в столбце C_i. Случай $\sum_{j=1}^{N_i} F_{ij} \geq (1 - \epsilon_4)|R|$ означает, что распределение перекошено, поэтому необходимо контейнеризировать значения в выборке путём объединения групп значений в категории и удаления из выборки строк, не содержащих самых частых значений столбца. Количество различных значений в i-м столбце d_i устанавливаем равным N_i. Если же распределение не перекошено, устанавливаем $d_i = min(|C_i|_R, d_{max})$, где R — исходная таблица.

FUNCTION `Category`$(i, x, d, SKEW)$**:**
IF SKEW = TRUE
THEN RETURN j such that $x = V_{ij}$
ELSE RETURN $1+[Hash(x) \bmod d]$.

Рис. 2: функция распределения значений по категориям, где V_{ij} — j-е самое частое значение в i-м столбце. Листинг взят из статьи [1].

Шаг 5, чтобы проверить столбцы на наличие корреляции строится таблица сопряженности f_{jk} размера $d_1 \times d_2$, а также вектора f_{j+} и f_{+k} размера d_1 и d_2 соответственно, содержащие суммы значений таблицы f_{jk} по строкам и по столбцам. Если в таблице сопряженности не слишком много структурных нулей (что свидетельствует о наличии корреляции), вычисляется коэффициент χ^2 по формуле[3]:

$$\chi^2 = \sum_{j=1}^{d_1} \sum_{k=1}^{d_2} \frac{(f_{jk} - \frac{f_{j+} f_{+k}}{n})^2}{f_{j+} f_{+k}}, \tag{2}$$

где n — итоговый размер выборки после фильтрации. χ^2 сравнивается с величиной $t = G_\nu^{-1}(1-p)$, где $\nu = (d_1 - 1)(d_2 - 1)$ — число степеней свободы, G_ν^{-1} — квантиль распределения хи-квадрат. Случай $\chi^2 > t$ означает наличие корреляции.

[3]Формула представленная в статье [1] неверна. Формула 2 взята из [5].

3. Реализация алгоритма

Алгоритм реализован на языке программирования C++ с применением стандартной библиотеки и собрания библиотек Boost.

На момент первоначальной реализации алгоритм состоит из двух классов: Cords и Sample. Реализация доступна по ссылке[4].

Класс Sample состоит из конструктора и вспомогательных полей. Конструктор использует класс uniform_int_distibution из стандартной библиотеки <random> [3], для того чтобы сформировать выборку с заменой данного размера из исходной таблицы, параллельно вычисляя число различных значений в столбцах выборки и их конкатенации.

Класс Cords разбит на 10 основных функций и несколько вспомогательных:

- GetFrequentValuesStatistics — для каждого столбца таблицы определяет число различных значений в нём и частоту появления в столбце для каждого значения, формируя unordered_map в котором ключами являются величины из столбца, а значениями пары вида (частота появления в столбце; порядковый номер, если упорядочить значения в столбце по частоте появления).
- ComputeSampleSize — вычисляет размер выборки по формуле 1.
- DetectSfd — определяет наличие МФЗ по процедуре, описанной в шаге 3.
- SkewHandling — проверяет перекошено ли распределение в данных столбцах, если перекошено, изменяет число категорий и вызывает Filter.
- Filter — удаляет из полученной выборки все строки, содержащие редко встречающиеся значения.
- Category — распределяет оставшиеся значения по категориям в соответствии с процедурой описанной на листинге 2.

[4] https://github.com/egshnov/DemoCORDS

- ComputeContingencyTable — формирует таблицу сопряженности для выборки.
- TooMuchStructuralZeroes — определяет число структурных нулей в таблице, если их слишком много, то корреляция обнаружена и алгоритм прекращает работу.
- ComputeChiSquared — вычисляет коэффициент χ^2 по формуле 2.
- ExecuteInternal — перебирает всевозможные пары столбцов таблицы, для каждой пары последовательно вызывает вышеописанные процедуры в соответствии с разделом 2.4.

4. Эксперименты

В данном разделе исследуется влияние входных данных на время работы алгоритма, соответствие предложенной реализации исходной статье [1], а также приводится сравнение времени выполнения алгоритма CORDS с алгоритмом Pyro реализованным в платформе DESBORDANTE.

Характеристики системы, на которой производились эксперименты: AMD Ryzen 5 5500U CPU @ 2.1GHz × 6, 8GiB RAM, Ubuntu 23.04.

4.1. Влияние параметров алгоритма на размер выборки

Наиболее существенным этапом алгоритма CORDS с точки зрения влияния на производительность является построение выборки поступившей таблицы, так как все последующие процедуры занимаются её обработкой. Соответственно, перед тем как приступать к замерам времени выполнения, требуется изучить каким образом меняет своё значения величина n (формула 1) в зависимости от параметров алгоритма.

Для исследования зависимости функция ComputeSampleSize (раздел 2.3) была запущена 40 тысяч раз с различными конфигурациями параметров. На рис. 3 представлен график зависимости размера выборки от максимальной допустимой вероятности ложно-положительного результата теста на корреляцию (параметр p, раздел 2.3), при фиксированных $d_1 = 644770, d_2 = 50, \delta = 0.05$.

Аналогичная картина наблюдается и при других значениях d_1, d_2, δ, что свидетельствует о том, что размер выборки требуемый для теста на наличие корреляции зависит не столько от размера исходной таблицы, сколько от её содержания и требуемой точности [1].

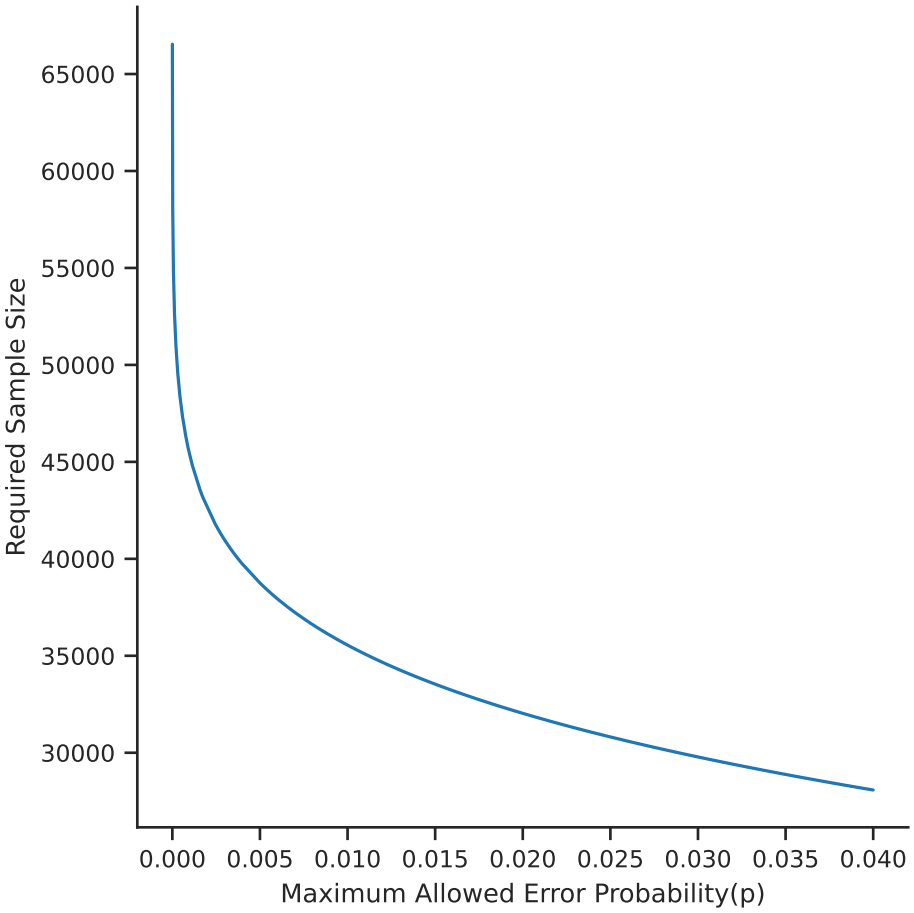

Рис. 3: Зависимость размера выборки от параметра p.

4.2. Зависимость времени выполнения от размера выборки

На рис. 4 представлен график зависимости времени выполнения алгоритма в миллисекундах от размера выборки. По оси ординат откладывается время выполнения алгоритма изображенного на рис. 1, а не всего алгоритма CORDS. В качестве входной таблицы использовалась таблица LINEITEM из TPC-H benchmark, содержащая 16 столбцов и около 726 тысяч строк. Время выполнения функции GetFrequentValuesStatistics из раздела 3 не учитывалось. Для получения статистики алгоритм был запущен 8 тысяч раз с различной конфигурацией параметров. Как видно из графика, время выполнения возрастает с ростом размера выборки в манере схожей с линейной, что близко к результатами получен-

ными в статье [1].

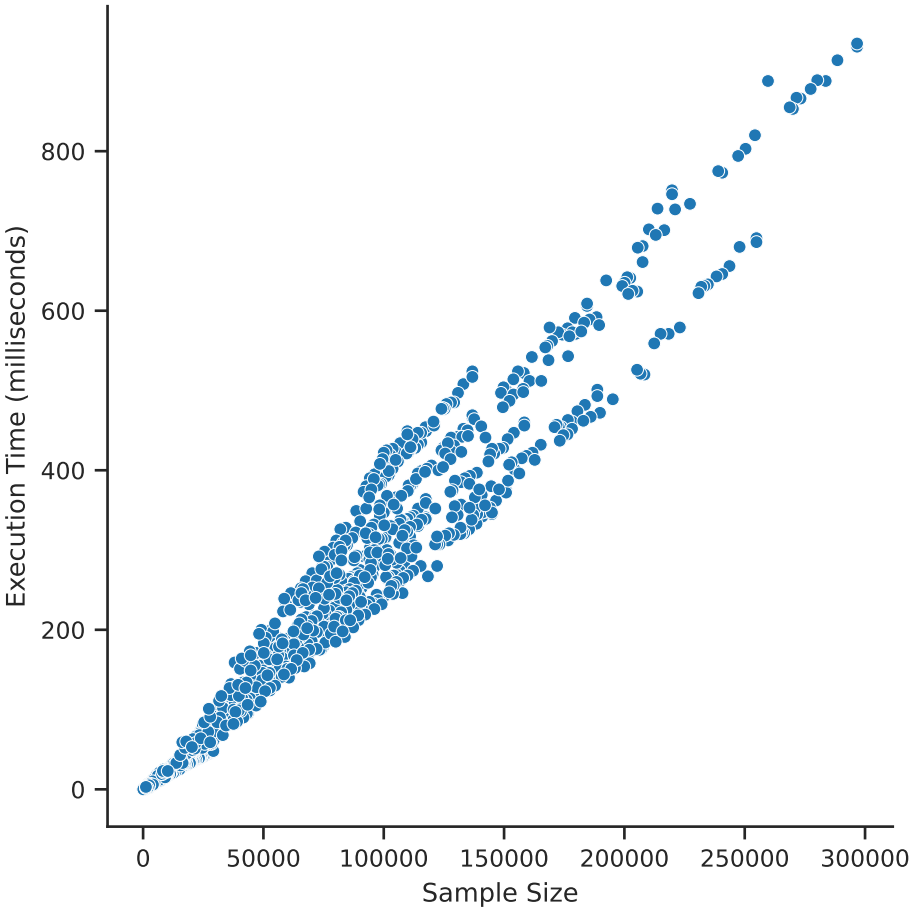

Рис. 4: Зависимость времени выполнения от размера выборки.

4.3. Сравнение времени выполнения с интегрированным алгоритмом

Для сравнения был выбран алгоритм Pyro, реализованный в платформе DESBORDANTE, так как он позволяет регулировать количество атрибутов в левой части искомой функциональной зависимости, сравнение с алгоритмами не позволяющими этого сделать некорректно, в связи с тем что CORDS выявляет зависимости лишь с одним атрибутом в левой части.

Таблица 2 содержит информацию о наборах данных, которые использовались в ходе эксперимента, а также среднее время исполнения для алгоритмов CORDS и Pyro в миллисекундах.

Таблица 2: Статистика по времени выполнения

Данные	Атрибуты	Строки	CORDS мс	Pyro мс
abalone	9	4177	24.7±0.29	24.4±0.32
adult	15	32561	147.3±0.65	52.4±0.88
breast cancer	30	570	15.9.4±0.35	46.7±0.7
CIPublicHighway	18	427511	128.5±0.52	286.4±5.3
CIPublicHighway10k	18	10000	38.6±0.59	32.8±0.49
digits	64	1798	477.1±1.67	1631.3±30.3
EpicMeds	10	1281732	1017.9±5.36	1652.6±30.5
EpicVitals	7	1246304	45.36±0.41	650.2±33.1
iowa1kk	24	1000001	139.2±0.56	2779.5±22.5
iris	5	150	2.6±0.6	2.7±0.21
LegacyPayors	4	1465234	240.81±1.3	512.1±7.1
lineitem	16	726360	2450.9±27.3	1351.3±51.6
lineitem537	16	537	36.1±0.19	22.6±0.49
neighbors10k	7	10000	28.4±0.59	7.6±0.14
neighbors100k	7	100000	83.2±0.48	22.2±0.6
WDC satellites	8	174	4.9±0.19	4.9±0.1

Каждый набор данных был обработан по 10 раз обоими алгоритмами. В результате экспериментов по полученным данным были построены 95% доверительные интервалы для среднего. Время необходимое для загрузки данных в систему не учитывалось. Pyro запускался с параметром $max\_lhs = 1$, определяющим количество атрибутов в левой части ФЗ. CORDS запускался с параметром p, равным 0.001.

Полученные результаты оказались противоречивыми, оба алгоритма на какой-то части данных существенно обходят оппонента, это может быть связано со спецификой алгоритма Pyro, являющегося рандомизированным. Для корректного сопоставления производительности требуется дальнейшее исследование.

Заключение

В ходе работы были выполнены следующие задачи:

- был написан обзор алгоритма CORDS;
- был реализован алгоритм нахождения мягких функциональных зависимостей и корреляций;
- было произведено тестирование алгоритма

Можно выделить следующие направления продолжения работы:

- реализация более эффективной версии алгоритма;
- интегрирование алгоритма в платформу Desbordante;
- проведение более подробного теста производительности;

Код алгоритма доступен на GitHub[5].

[5] https://github.com/egshnov/DemoCORDS

Список литературы

[1] CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies / Ihab F. Ilyas, Volker Markl, Peter Haas et al. // Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. — SIGMOD '04. — New York, NY, USA : Association for Computing Machinery, 2004. — P. 647–658. — URL: https://doi.org/10.1145/1007568.1007641.

[2] Correlation Maps: A Compressed Access Method for Exploiting Soft Functional Dependencies / Hideaki Kimura, George Huo, Alexander Rasin et al. // Proc. VLDB Endow. — 2009. — aug. — Vol. 2, no. 1. — P. 1222–1233. — URL: https://doi.org/10.14778/1687627.1687765.

[3] Cppreference. Справочник по стандартной библиотеке C++. — [Online; accessed 2024-01-04]. URL: https://en.cppreference.com/w/.

[4] Chernishev George, Polyntsov Michael, Chizhov Anton et al. Desbordante: from benchmarking suite to high-performance science-intensive data profiler (preprint). — 2023. — 2301.05965.

[5] Hays W.L. 18.3 Pearson chi-square test for association // Statistics. — Harcourt Brace College Publishers, 1994. — P. 856–861. — URL: https://books.google.ru/books?id=zSi2AAAAIAAJ.

[6] Song Shaoxu, Chen Lei. Categorical Data // Integrity Constraints on Rich Data Types. — Cham : Springer International Publishing, 2023. — P. 15–46. — ISBN: 978-3-031-27177-9. — URL: https://doi.org/10.1007/978-3-031-27177-9_2.

[7] Unidata. Data profiling, и с чем его едят. — [Online; accessed 2024-01-04]. URL: https://habr.com/ru/companies/unidata/articles/667636/.