Санкт-Петербургский государственный университет

Кафедра иноформационно-аналитических систем

Группа 22.Б07-мм

Реализация мер приближенных функциональных зависимостей в проекте DESBORDANTE

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 представляет собой систематический процесс анализа и оценки характеристик наборов данных с целью выявления их структуры, качества и соответствия установленным требованиям. Эта методология включает в себя применение статистических и аналитических инструментов для извлечения метаданных, что позволяет обнаруживать аномалии, пропущенные значения и закономерности, скрытые в данных.

Важным примером таких закономерностей являются функциональные зависимости в базах данных. Функциональная зависимость вида $X \to Y$ означает, что если две записи таблица равны по атрибутам X, то они должны быть равны по атрибутам Y.

Однако, реальные данные часто содержат различные дефекты из-за которых точные зависимости не удерживаются, и поисковые алгоритмы не могут их обнаружить. В таких случаях требуется ослабить определение функциональной зависимости. Существуют различные приближения функциональных зависимостей, основанные на мерах силы (ошибки) зависимости, вычисляющих долю записей, удовлетворяющих (нарушающих) зависимость. В исследовательских работах часто встречаются приближенные функциональные зависимости основанные на модифицированной мере g_1 [3]. Впрочем, помимо g_1 существует целый ряд не менее интересных мер, например $pdep, \tau, \mu+, \rho$ и другие [4]. Более того, в статье [4] показывается, что мера g_1 является наименее подходящей для выявления приближенных функциональных зависимостей, с точки зрения точности выявления, в то время как мера μ^+ является одной из наиболее подходящих.

Desbordante [1] — высокопроизводительный инструмент для профилирования данных с открытым исходным кодом[1], разработанный с использованием языка программирования C++. Он способен выявлять множество закономерностей в данных, в том числе точные функциональные зависимости и приближенные функциональные зависимо-

[1] https://github.com/Desbordante/desbordante-core

сти основанные на мере g_1. Данная работа посвящена модификации DESBORDANTE позволяющей выявлять приближенные функциональные зависимости, основанные на мерах $pdep, \tau, \mu+, \rho$.

1 Постановка задачи

Целью данной работы является расширение профилировщика данных DESBORDANTE с целью поддержки выявления приближенных функциональных зависимостей, основанных на мерах $pdep, \tau, \mu+, \rho$. Для её выполнения были поставлены следующие задачи:

- провести обзор предметной области;
- реализовать выявление приближенных зависимостей на основе данных мер в платформе DESBORDANTE;
- экспериментально исследовать производительность реализации.

2 Обзор

2.1 Точные и ослабленные зависимости

Большинство определений и нотация взяты из статьи [4].

Определение 1. *Функциональная зависимость, это выражение вида* $\varphi := \mathbf{X} \to \mathbf{Y}$. *Отношение* $R(\mathbf{W})$, *где* $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{W}$ *и* $\mathbf{X} \cap \mathbf{Y} = \emptyset$ *удовлетворяет функциональной зависимости* φ, *если* $\forall\, \mathbf{w}, \mathbf{w}' \in R$ *мы имеем, что* $\mathbf{w}|_{\mathbf{Y}} = \mathbf{w}'|_{\mathbf{Y}}$ *при* $\mathbf{w}|_{\mathbf{X}} = \mathbf{w}'|_{\mathbf{X}}$.

Определение 2. *Мерой силы приближенной функциональной зависимости называется функция* $f := (\varphi, R) \to [0, 1]$, *вычисляющая на каком уровне функциональная зависимость* φ *удерживается в* R *(какую долю* R *составляют кортежи, чьё удаление приведет к тому, что зависимость будет удовлетворять определению 1).*

Таблица 1: Обозначения используемые в данной работе

X, Y	атрибуты
$dom(X)$	домен атрибута X
$\mathbf{X}, \mathbf{Y}$	конечные множества атрибутов
$\mathbf{x}$	кортеж на множестве атрибутов $\mathbf{X}$
$\mathbf{x} : \mathbf{X}$	x является кортежом на $\mathbf{X}$
$\mathbf{x}\|_{\mathbf{Y}}$	сужение $\mathbf{x}$ на $\mathbf{Y}$, где $\mathbf{Y} \subseteq \mathbf{X}$
$\mathbf{XY}$	$\mathbf{X} \cup \mathbf{Y}$
$\mathbf{xy}$	то же для кортежей, т.е. $\mathbf{xy}\|_{\mathbf{X}} = \mathbf{x}$ и $\mathbf{xy}\|_{\mathbf{Y}} = \mathbf{y}$
R	отношение
$R(\mathbf{X})$	R является отношением на X
$R(\mathbf{x})$	частота $\mathbf{x}$ в R
$\mathbf{x} \in R$	$\mathbf{x}$ является кортежем в R с $R(\mathbf{x}) > 0$
$dom_R(\mathbf{Y})$	$\{(x)\|_{\mathbf{Y}} : x \in R\}$
$\|R\|$	Общее число кортежей в R т.е. $\sum_{\mathbf{x}:\mathbf{X}} R(\mathbf{x})$
$\varphi := \mathbf{X} \to \mathbf{Y}$	функциональная зависимость

Чем меньше кортежей в R нарушают функциональную зависимость φ, тем выше значение f. Также, мы требуем чтобы $f(\varphi, R) = 1$, если R полностью удовлетворяет φ.

Замечание. *Во многих работах, например [6], определяется не мера силы, а ошибка функциональной зависимости e, в таком случае мера определяется как* $f_e(\varphi, R) = 1 - e(\varphi, R)$. *В данной работе мы считаем термины мера силы и мера ошибки равносильными.*

Ниже представлены формулы мер, исследуемых в данной работе.

Определение 3. *Вероятностью вхождения кортежа w назовём*

$$p_R(\mathbf{w}) := \frac{R(\mathbf{w})}{|R|}.$$

Определение 4. *Мерой* g_1 *назовём*

$$g_1(\mathbf{X} \to \mathbf{Y}, R) := 1 - \sum_{\mathbf{x},\mathbf{y}} p_R(\mathbf{xy})[p_R(\mathbf{x}) - p_R(\mathbf{xy})].$$

Определение 5. *Мерой* ρ *назовём*

$$\rho(\mathbf{X} \to \mathbf{Y}, R) := \frac{|dom_{\mathbf{X}}(R)|}{|dom_{\mathbf{XY}}(R)|}.$$

Определение 6. *Пусть* $dom(\mathbf{Y}) = \{1, \ldots, M\}$ *с частотами* $y_1, \ldots, y_M$, $dom(\mathbf{X}) = \{1, \ldots, K\}$, *с частотами* $x_1, \ldots, x_K$, $|R| = N$, *функцией самозависимости (self-dependency measure) [5] назовём*

$$pdep(\mathbf{Y}, R) := \sum_{j=1}^{M} \frac{y_j^2}{N^2}.$$

Определение 7. *В терминах определения 6 обозначим число вхождений объединений значений атрибутов за* $n_{ij} = |\{w \in R : w|_{\mathbf{X}} = i \wedge w|_{\mathbf{Y}} = j\}|$, *мерой pdep назовём*

$$pdep(\mathbf{X} \to \mathbf{Y}, R) := \frac{1}{N} \sum_{i=1}^{K} \sum_{j=1}^{M} \frac{n_{ij}^2}{x_i}.$$

Определение 8. *Мерой* τ *назовём*

$$\tau(\mathbf{X} \to \mathbf{Y}, R) := \frac{pdep(\mathbf{X} \to \mathbf{Y}, R) - pdep(\mathbf{Y}, R)}{1 - pdep(\mathbf{Y}, R)}.$$

Определение 9. *В терминах определения 6 обозначим за μ функцию*

$$\mu(\mathbf{X} \to \mathbf{Y}, R) := 1 - \frac{1 - pdep(\mathbf{X} \to \mathbf{Y}, R)}{1 - pdep(\mathbf{Y}, R)} \times \frac{N-1}{N-K},$$

мерой μ^+ назовём

$$\mu^+(\mathbf{X} \to \mathbf{Y}, R) := max(\mu(\mathbf{X} \to \mathbf{Y}, R), 0).$$

Замечание. *Функция μ сама по себе не является мерой так как может принимать отрицательные значения [4].*

2.2 Алгоритмы выявления приближенных зависимостей

При переходе к реализации встаёт вопрос о выборе алгоритма, который будет заниматься выявлением зависимостей. В ходе научного поиска было выделено два основных алгоритма обнаружения приближенных функциональных зависимостей.

2.2.1 Алгоритм Tane

Tane является одним из первых алгоритмов позволяющих находить все нетривиальные минимальные приближенные функциональные зависимости на основе меры g_3 (реализация представленная в DESBORDANTE применяет меру g_1).

Алгоритм организует пространство поиска в виде своеобразной решетки, состоящей из атрибутов потенциальных зависимостей, соответствующих зависимостям вида $X \backslash \{A\} \to A$. Решетка разделена на уровни по размеру левой части.

В целях повышения эффективности поиска зависимостей применяется 3 основных правила отсечения: отсечение ключей и суперключей, а также 2 вариации отсечения зависимостей не являющихся минимальными по отношению к уже выявленным зависимостям. Важно отметить, что корректность этих процедур не зависит от используемой меры ошибки [6], что позволяет свободно применять меры отличные от

g_1 и g_3. Более того, похожая работа уже была проделана с примитивом PFD [2].

2.2.2 Алгоритм Pyro

Алгоритм Pyro на данный момент является самым эффективным алгоритмом выявления приближенных функциональных зависимостей и существенно превосходит Tane по производительности [3].

Алгоритм делит пространство поиска на решетки с фиксированной правой частью потенциальной зависимости, элементы решетки представляют собой множества атрибутов, входящих в левую часть потенциальной зависимости. Pyro начинает поиск с зависимостей, содержащих один атрибут в левой части, постепенно расширяя её. В процессе расширения Pyro заведомо отсекает зависимости, не являющиеся минимальными, и зависимости, чья ошибка оказывается выше заданного параметра. Описанный процесс существенно полагается на свойство монотонности меры g_1. Кроме того, Pyro опирается на возможность приблизительно оценивать значение g_1 зависимости-кандидата без явного её вычисления, что существенно ускоряет алгоритм. Эта возможность была обоснована в лемме 1 статьи [3]. Таким образом, для того чтобы напрямую внедрить новую меру внутрь Pyro необходимо:

- доказать монотонность новой меры, и,
- доказать лемму 1.

Такая серьёзная привязанность алгоритма к мере силы не позволяет применять его для других мер без серьёзной подготовки (доказательств) или же большой переработки алгоритма, которая может вылиться в создание нового.

2.2.3 Вывод

На первый взгляд алгоритм Pyro может показаться более подходящим вариантом в виду своей эффективности, однако близкое рассмотрение показывает, что внедрение новых мер зависимостей в Pyro суще-

ственно осложняется необходимостью 1) доказать монотонность каждой меры, 2) представить процедуру аналогичную предсказанию значения g_1. Фокус данной работы не на реализации наиболее эффективного алгоритма, а создание первого прототипа данной функциональности в рамках DESBORDANTE. Наша конечная цель — предоставить пользователю возможность искать зависимости с данными мерами, хотя бы как-то. Так как описанные выше проблемы существенно осложняют внедрение в Pyro, было принято решение реализовать меры $pdep, \rho, \tau, \mu^+$ на базе Tane.

3 Решение

На момент начала работы в проекте DESBORDANTE уже был реализован алгоритм Tane, выявляющий приближенные функциональные зависимости основанные на мере g_1. Более того в работе [2] он был разделен на класс-родитель TaneCommon, содержащий всю основную логику алгоритма кроме вычисления меры силы потенциальной зависимости, и дочерний класс Tane, реализующий недостающие функции.

При внедрении новых мер вместо того, чтобы выносить каждую новую меру в отдельный класс, было принято решение модифицировать класс Tane параметром error_measure (со значением по умолчанию g_1), указывающим на применяемую меру силы. Оставшаяся работа заключалась в адаптации формул, представленных в разделе 2.1 к структурам, применяемым в алгоритме Tane. Опишем их в терминах работы [6].

Определение 10. *В терминах таблицы 1 обозначим класс эквивалентности по множеству атрибутов* $\mathbf{X}$ *как* $[t]_{\mathbf{X}} = \{u \in R : t|_A = u|_A \, \forall \, A \in X\}$.

Определение 11. *Множество* $\pi_{\mathbf{X}}$ *классов эквивалентности по атрибутам* $\mathbf{X}$*, где* $\pi_{\mathbf{X}} = \{[t]_X : t \in R\}$ *назовем партицией (Partition).*

Определение 12. *Партицию, из которой удалили классы эквивалентности размера 1, назовём сокращенной партицией (Stripped Partition) и обозначим за* $\bar{\pi}_{\mathbf{X}}$.

Определение 13. *Сумму мощностей классов эквивалентности, входящих в сокращенную партицию* $\bar{\pi}_{\mathbf{X}}$*, назовем объёмом партиции и обозначим как* $||\bar{\pi}_{\mathbf{X}}||$.

Tane вычисляет значение меры ошибки на основе сокращенных партиций. Ниже представлены алгоритмы вычисления мер самозависимости (6) и *pdep*, оставшиеся меры очевидным образом вычисляются на основе формул, представленных в разделе 2.1

Также было произведено модульное тестирование кода с использованием библиотеки Googletest. Новая функциональность автоматически выведена в python библиотеку DESBORDANTE.

Algorithm 1 Алгоритм вычисления меры самозависимости

Input: Отношение R, партиция $\bar{\pi}_Y$

procedure CalculatePdepSelf($R, \bar{\pi}_Y$)
    $N \leftarrow |R|$
    $S \leftarrow 0$
    $k \leftarrow 0$
    for *each* $[t]_{\mathbf{Y}} \in \bar{\pi}_{\mathbf{Y}}$ **do**
        $n \leftarrow k + |[t]_{\mathbf{Y}}|$
        $S \leftarrow S + |[t]_{\mathbf{Y}}|^2$
    end for
    $S \leftarrow S + N - k$
    return $\frac{S}{N^2}$
end procedure

Algorithm 2 Алгоритм вычисления меры pdep

Input: Отношение R, партиция $\bar{\pi}_{XA}$

procedure CalculatePdep($R, \bar{\pi}_{XA}$)
    $N \leftarrow |R|$
    $S \leftarrow 0$
    for *each* $[t]_{\mathbf{XA}} \in \bar{\pi}_{\mathbf{XA}}$ **do**
        $n \leftarrow |[t]_{\mathbf{XA}}|^2$
        $d \leftarrow R(t|_{\mathbf{X}})$
        $S \leftarrow S + \frac{n}{d}$
    end for
    for *each* $t \notin \bar{\pi}_{\mathbf{XA}}$ **do**
        $S \leftarrow S + \frac{1}{R(t|_{\mathbf{X}})}$
    end for
    return $\frac{S}{N}$
end procedure

4 Эксперименты

Характеристики системы, на которой производились эксперименты: AMD Ryzen 5 5500U CPU @ 2.1GHz (6 cores), 8GiB RAM, Ubuntu 23.04, gcc 13.2.0, C++20.

В таблице 3 представлены результаты сравнения производительности мер на наборах данных из таблицы 2. Исходные данные доступны по ссылке[2]. На диаграмме 1 представлена визуализация замеров на примере таблицы CIPublicHighway200k (аналогичная картина наблюдается и на других наборах данных).

Каждый набор данных был обработан по 40 раз. В результате экспериментов по полученным данным были построены 95% доверительные интервалы для среднего. Время необходимое для загрузки данных в систему не учитывалось. Во избежание влияния обрезки пространства поиска на время выполнения, параметру $error\_measure$ было присвоено значение 0. Перед проведением замеров, были проделаны следующие действия: отключены все фоновые процессы, установлена максимальная частота процессора, выключен swap, сброшен дисковый кэш. Также был совершен прогрев из 10 проходов, программа была исполнена одним ядром.

Как видно из диаграммы 1, мера g_1 занимает первое место по времени исполнения, второе место занимает мера ρ, в то время как меры, основанные на $pdep$, ввиду своей вычислительной сложности, являются наименее эффективными. Это связано с тем, что все данные необходимые для вычисления g_1 доступны напрямую из сокращенной партиции, в то время как для вычисления $pdep$ и ρ требуются дополнительные действия, меры τ, μ^+ вычисляются по формулам из раздела 2.1, из-за чего их производительность практические не отличается от $pdep$.

[2] https://github.com/Desbordante/desbordante-core/tree/main/datasets

Таблица 2: Свойства таблиц

Данные	Атрибуты	Строки
CIPublicHighway100k	18	100K
CIPublicHighway200k	18	200K
neighbors10k	7	10K
neighbors100k	7	10K
WDCsatellites	8	173

Таблица 3: Сравнение производительности в миллисекундах

Данные	g_1	$pdep$	τ	μ^+	ρ
CIPublicHighway100k	2355.0 ± 8.1	6359.0 ± 6.8	6389.4 ± 7.4	6404.6 ± 7.7	3731.1 ± 4.9
CIPublicHighway200k	4494.4 ± 479.1	13893.7 ± 1687.5	13908.0 ± 1683.7	14087.2 ± 1720.5	7144.7 ± 764.5
neighbors10k	5.8 ± 0.03	11.2 ± 0.02	11.3 ± 0.04	11.3 ± 0.02	8.0 ± 0.02
neighbors100k	40.2 ± 7.7	80.7 ± 15.6	81.1 ± 15.6	81.5 ± 15.7	58.6 ± 11.3
WDCsatellites	26.9 ± 6.1	54.1 ± 12.4	54.3 ± 12.4	54.6 ± 12.5	39.3 ± 8.9

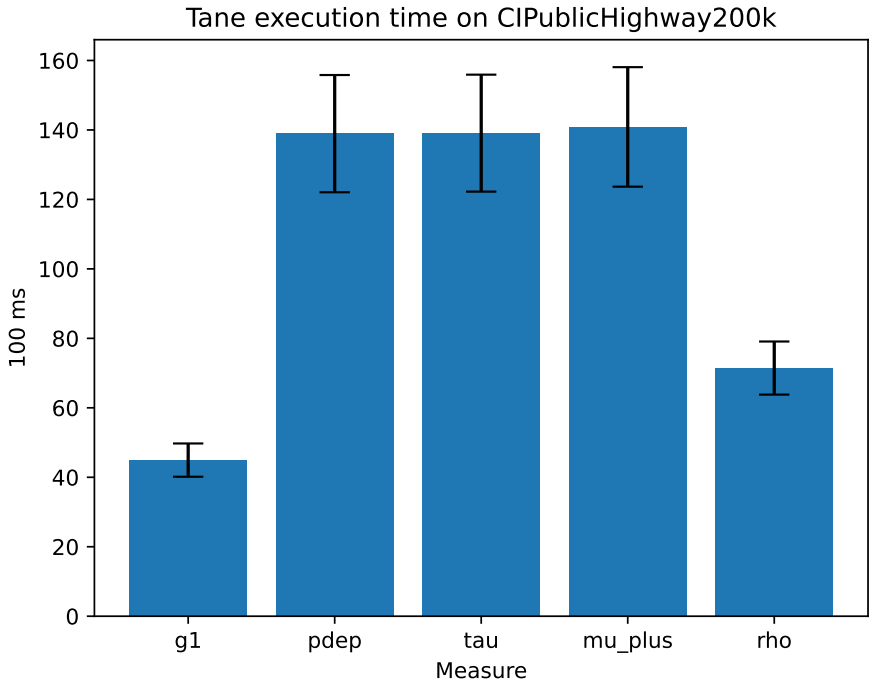

Рис. 1

Заключение

В ходе работы были выполнены следующие задачи:

- был проведен обзор предметной области;
- было реализовано выявление приближенных функциональных зависимостей на основе мер $pdep, \tau, \mu+, \rho$;
- было произведено экспериментальное исследование производительности.

Можно выделить следующие направления продолжения работы:

- реализовать оставшиеся меры представленные в статье [4];
- воспроизвести результаты исследования мер из статьи [4].

Код доступен по ссылке на GitHub[3].

[3] https://github.com/Desbordante/desbordante-core/pull/458

Список литературы

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

[2] Extending Desbordante with Probabilistic Functional Dependency Discovery Support / Ilia Barutkin, Maxim Fofanov, Sergey Belokonny et al. // 2024 35th Conference of Open Innovations Association (FRUCT). — 2024. — P. 158–169.

[3] Kruse Sebastian, Naumann Felix. Efficient discovery of approximate dependencies // Proc. VLDB Endow. — 2018. — mar. — Vol. 11, no. 7. — P. 759–772. — URL: https://doi.org/10.14778/3192965.3192968.

[4] Measuring Approximate Functional Dependencies: A Comparative Study / Marcel Parciak, Sebastiaan Weytjens, Niel Hens et al. // 2024 IEEE 40th International Conference on Data Engineering (ICDE). — 2024. — P. 3505–3518.

[5] Piatetsky-Shapiro Gregory, Matheus Christopher J. Measuring data dependencies in large databases // Proceedings of the 2nd International Conference on Knowledge Discovery in Databases. — AAAIWS'93. — AAAI Press, 1993. — P. 162–173.

[6] TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies / Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, Hannu Toivonen // Comput. J. — 1999. — Vol. 42, no. 2. — P. 100–111. — URL: https://doi.org/10.1093/comjnl/42.2.100.