Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Щукин Илья Вячеславович

Реализация инфраструктуры для построения датасетов машинного обучения для предсказания времени работы алгоритмов поиска функциональных зависимостей

Отчёт по учебной практике

Научный руководитель:
ассистент кафедры ИАС Чернышев Г. А.

Санкт-Петербург
2022

Оглавление

1 Введение

Как сказано в статье [4] функциональные зависимости являются важными метаданными. Они показывают зависимости между атрибутами отношения базы данных. Пусть r — отношение, X и Y — произвольные подмножества множества атрибутов отношения r. Тогда Y функционально зависит от X ($X \to Y$) тогда и только тогда, когда каждое значение X связано точно с одним значением множества Y [2]. Функциональные зависимости применяются для нормализации баз данных, для фильтрации и анализа данных. Для поиска функциональных зависимостей существует множество различных алгоритмов [4].

Время работы алгоритмов поиска функциональных зависимостей сложным образом зависит от свойств входных данных и характеристик ЭВМ, например: количества функциональных зависимостей или типов колонок. Эвристическими методами невозможно аппроксимировать время с достаточной точностью. При этом существует оценки времени важна для практических задач. Из-за чего возникает необходимость применения машинного обучения для выделения закономерностей, влияющих на производительность.

В данной работе предлагается инфраструктура для сбора данных, необходимых для обучения модели предсказания времени работы алгоритмов поиска функциональных зависимостей. Данная задача требует возможности простой переносимости и развёртывания для чего были разработаны докер-образ и телеграм-бот. Также проводится классификация признаков которые понадобятся для обучения. Дальше будут представлены собранные датасеты и их характеристики с учетом запусков на Pyro [5].

2 Постановка задачи

Целью данной работы является разработка тулкита для сбора данных необходимых для обучения модели машинного обучения для предсказания времени работы алгоритмов поиска функциональных зависимостей. Для её выполнения были поставлены следующие задачи:

1. Выделить критерии которые могут влиять на время работы, классифицировать их.
2. Создать докер-образ для развертывания сборщика статистики на различных машинах.
3. Написать телеграм-бота для сбора и агрегации информации по результатам запуска.
4. Собрать датасеты-кандидаты и систематизировать их.

3 Обзор

3.1 Metanome

Metanome [1] — платформа для алгоритмов поиска функциональных зависимостей, разработанная институтом Хассо Платтнера и институтом исследования вычислений. Metanome является первой созданной платформой в этой области, долгое время он применялся для тестирования существующих и создания новых алгоритмов. Основным языком разработки бэкенда и всех алгоритмов является Java, что негативно влияет на производительность вычислений.

3.2 Desbordante

Для того, чтобы разрешить проблемы с производительностью алгоритмов был создан проект Desbordante [3]. Он также является платформой для датамайнинга, но для разработки используется язык C++. В данной работе используются реализации алгоритмов из Desbordante для получения времён работы. Из таблицы ?? видно, что Desbordante производительнее, именно поэтому его использование предпочтительнее для данной работы.

Таблица 1: Сравнение реализаций Desbordante и Metanome

Implementation	adult	breast_cancer	CIPublicHighway	EpicMeds
Desbordante	8381 ± 154	34 ± 0	166342 ± 156446	24599 ± 873
Metanome	8177 ± 176	117 ± 2	210615 ± 198689	55680 ± 4346

Implementation	EpicVitals	Iowa1KK	LegacyPayors	Neighbors100K
Desbordante	2580 ± 33	22854 ± 566	385 ± 21	35 ± 3
Metanome	3383 ± 103	27228 ± 318	875 ± 44	122 ± 8

3.3 FD_Mine

Алгоритм FD_Mine [9] состоит из 2 фаз: поиск зависимостей и восстановление ответа. Во время поиска зависимостей на каждой итерации

рассматривается набор кандидатов, кандидатом является набор атрибутов, который может стоять в левой части функциональной зависимости. Для каждого кандидата строится замыкание и находятся функциональные зависимости с ключами. Дальше находятся все эквивалентности для набора кандидатов. На следующем этапе алгоритм отсеивает кандидатов, для которых находятся эквивалентные. В конце каждой итерации строится новый набор кандидатов.

Во время реализации алгоритма для Desbordante [3] было выявлено, что время восстановления ответа очень чувствительно к количеству функциональных зависимостей в данных. Что показывает, что время выполнения алгоритмов может сильно зависеть от внутреннего устройства самих датасетов. Точное предсказание времени работы алгоритмов требует анализа закономерностей и свойств данных.

3.4 Современные алгоритмы

Современные алгоритмы поиска функциональных зависимостей, такие как [5–7] требуют множества вычислительных ресурсов. При этом трудно оценить время их работы из-за сложного многоэтапного устройства некоторых из них и из-за факторов, связанных с данными и ЭВМ. Оценивать время работы необходимо для решения практических задач, связанных с выделением ресурсов и с планированием.

До данной работы не существовало подходящего набора датасетов и инструмента для сбора данных, которые позволили бы обучить модель машинного обучения.

4 Идея подхода

Для обучения предсказательной модели потребуется множество признаков: информация о вычислительной машине, данные метаданные из датасетов. В качестве метеданных можно выбрать: вес таблицы, количество колонок и строк и другие.

Также для расширения множества признаков получаемых из датасетов можно применить подход описанный в [8]. Авторы данной работы используют множество сгенерированных признаков для обучения модели машинного обучения. Глобальные статистики, например энтропия столбца, среднее числовых данных. Распределения символов, например отношение количества цифр ко всем символам, производные от количества символов ASCII (среднее, минимум, максимум и т.д.)

Данный метод позволит извлекать из датасетов значительное количество полезной для обучения информации. Что должно улучшить качество предсказаний модели.

5 Данные

Признаки для обучения модели можно разбить на два основных класса. Признаки связанные с ЭВМ и датасетами.

5.1 Выбор датасетов

При выборе датасетов необходимо было обеспечить наибольшее разнообразие возможных признаков (размер, типы колонок, количество столбцов и другие). Также необходимо было обеспечить равномерное покрытие интервала времён исполнения для алгоритма поиска функциональных зависимостей для исключения возможной смещённости.

Данные были взяты с сайтов: Kaggle, telecom-paristech, Rdatasets, HPI, Data.gov

Полученный набор датасетов можно увидеть в таблице 2

5.2 Признаки вычислительных машин

В качестве признаков, влияющих на время исполнения алгоритмов были выбраны следующие характеристики ЭВМ: модель процессора, количество ядер, объем оперативной памяти, частота памяти и частота процессора и объём кэша. Возможные признаки, связанные с жёсткими дисками были опущены, т.к. алгоритмы практически не взаимодействуют с ними, а значит их влияние на результат минимально.

Таблица 2: Полученный набор датасетов

name	time in sec	size in kb	columns	rows	fds
Expense_Actuals.csv	0.04	81	7	1488	9
Environmental_Conservation_Staff_Office_Locations.csv	0.12	23	11	143	47
Capital_Grant_Awards_2016.csv	0.17	42	14	200	130
neighbors100k.csv	0.40	6469	7	99999	12
LegacyPayors.csv	1.61	20602	1	1465233	0
EpicVitals.csv	8.89	33396	7	1246303	2
EpicMeds.csv	30.07	55448	10	1281731	16
dice_com-job_us_sample.csv	0.44	59875	12	22000	39
monster_com-job_sample.csv	0.45	66785	14	22000	22
naukri_com-job_sample.csv	0.61	51037	14	22000	94
r_dataisbeautiful_posts.csv	1.57	39990	12	190853	36
2019 - 01.csv	1.81	18972	10	192082	17
HN_posts_year_to_Sep_26_2016.csv	2.52	46250	7	293119	17
uci-news-aggregator.csv	3.10	100484	8	422419	19
covid19_tweets.csv	3.92	67103	13	179108	126
wetlands.csv	4.19	81962	7	670117	22
train-balanced-sarcasm.csv	8.75	249286	10	1010826	21
jena_climate_2009_2016.csv	11.77	42152	15	420551	1286
city_temperature.csv	14.52	137305	8	2906327	3
Food_Inspections.csv	17.19	180426	17	153810	2115
tiktok_app_reviews.csv	18.24	530295	5	3646476	5
FINAL_FROM_DF.csv	19.51	76266	13	846404	409
Bitcoin_tweets.csv	23.07	684406	13	1793124	10
votes.csv	26.41	337046	10	2933938	15
measures_v2.csv	30.69	293028	13	1330816	1251
bitstampUSD_1-min_data_2012-01-01_to_2021-03-31.csv	40.55	310084	8	4857377	23
TrendingTopics.csv	79.77	1028618	26	4899080	74028
final_tripdata.csv	178.95	1291180	16	5515094	349
london_crime_by_lsoa.csv	195.36	910940	7	13490604	3
database.csv	274.08	1938796	14	12157458	158

name	time in sec	size in kb	columns	rows	fds
chess_games.csv	323.05	4276264	15	6256184	597
balance-scale.csv	0.02	6	5	624	1
iris.csv	0.13	4	5	150	4
breast.csv	0.26	19	11	699	46
nursery.csv	0.26	1034	9	12959	1
chess.csv	0.27	519	7	28055	1
bridges.csv	0.28	6	13	108	142
abalone.csv	0.32	187	9	4177	137
echocardiogram.csv	0.55	5	13	131	527
adult.csv	0.73	3527	15	32560	78
ncvoter_1001r_19c.csv	1.55	150	19	1000	758
letter.csv	1.96	695	17	19999	61
hepatitis.csv	11.43	7	20	154	8296
fd-reduced-30.csv	67.23	69580	30	250000	89571
cps1.csv	0.11	739	11	15992	11
close_elections_lmb.csv	0.29	987	10	13588	30
mortgages.csv	0.58	7030	7	214144	7
Fertility.csv	1.02	12570	9	254654	8
military.csv	2.68	72675	7	1414593	6

6 Реализация тулкита

6.1 Docker

При использовании нескольких машин с разными ОС возникает проблема переносимости. Обеспечить переносимость помогла контейнеризация с помощью Docker, показанная на Рис. 1. Это значительно упрощает развёртывание консольного приложения Desbordante.

Сборка Docker-образа происходит в две стадии. На первой происходит установка всех необходимых зависимостей и компиляция Desbordante. На второй устанавливаются зависимости Python скрипта и сам скрипт. При этом из первой стадии сборки используется только готовый исполняемый файл Desbordante. Это позволило значительно уменьшить размер образа, что упрощает его передачу через интернет.

6.2 Отправка и агрегация данных

Для удобства и простоты сбора получаемых замеров было решено использовать телеграм-бота. Такой подход значительно упрощает агрегацию данных в сравнении с электронной почтой или передачей через ftp.

6.3 Реализация сборщика

Для создания программы ответственной за сбор данных был выбран язык Python. Скрипт запускает Desbordante поочередно на предоставленных ему датасетах. После окончания вычислений программа формирует csv таблицу и отправляет её через telegram-бота.

Также была учтена возможная проблема с недостаточным количеством ОЗУ, при которой oom killer мог останавливать всю ветку процессов, что приводило бы к потере данных. Эта проблема была решена с помощью ограничения виртуальной памяти доступной Desbordante через ulimit.

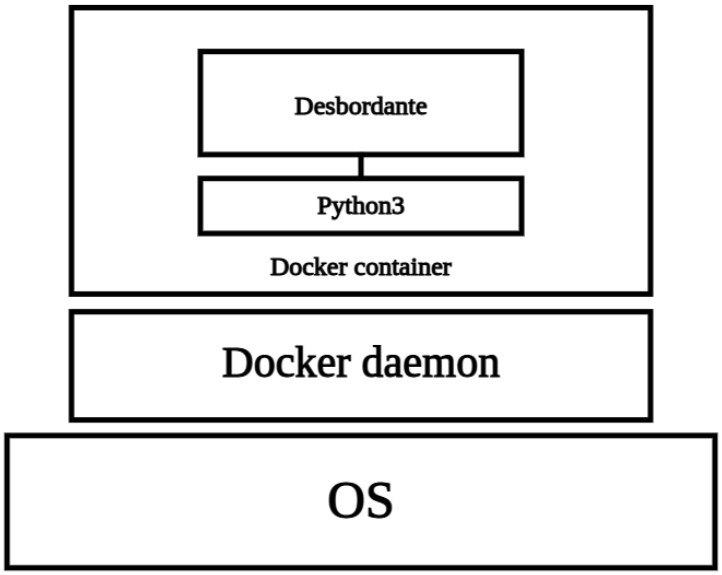

Рис. 1: Архитектура сборщика

7 Дальнейшие планы

- Собрать данные с ЭВМ.
- Проанализировать полученные данные. Выявить, какие из признаков оказывают большее влияние на время исполнения.
- Построить модель и оценить её предсказания.
- Сравнить модель с эвристическими подходами.
- Развернуть полученную модель.

8 Заключение

В ходе работы были получены следующие результаты:

- Выделены основные критерии влияющие на время работы.
- Разработан докер-образ для сборщика.
- Создан телеграм-бот для агрегации данных.
- Собран обширный набор датасетов.

Полученный набор датасетов с указанием источников доступен по ссылке[1].

Код тулкита можно найти на github.

[1] https://docs.google.com/spreadsheets/d/1zV3Ru0VKtdT9U4aZk3NXvnidzs5FZP11Avfmqdph9xA/edit?usp=sharing

Список литературы

[1] Papenbrock Thorsten, Bergmann Tanja, Finke Moritz, Zwiener Jakob, and Naumann Felix. Data Profiling with Metanome // Proc. VLDB Endow. — 2015. — Aug. — Vol. 8, no. 12. — P. 1860–1863. — Access mode: http://dx.doi.org/10.14778/2824032.2824086.

[2] Date C.J. An Introduction to Database Systems. — 8 ed. — USA : Addison-Wesley Longman Publishing Co., Inc., 2003. — ISBN: 0321197844.

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

[4] Papenbrock Thorsten, Ehrlich Jens, Marten Jannik, Neubert Tommy, Rudolph Jan-Peer, Schönberg Martin, Zwiener Jakob, and Naumann Felix. Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms // Proc. VLDB Endow. — 2015. — June. — Vol. 8, no. 10. — P. 1082–1093. — Access mode: https://doi.org/10.14778/2794367.2794377.

[5] Kruse Sebastian and Naumann Felix. Efficient Discovery of Approximate Dependencies // Proc. VLDB Endow. — 2018. — mar. — Vol. 11, no. 7. — P. 759–772. — Access mode: https://doi.org/10.14778/3192965.3192968.

[6] Papenbrock Thorsten and Naumann Felix. A Hybrid Approach to Functional Dependency Discovery // Proceedings of the 2016 International Conference on Management of Data. — New York, NY, USA : Association for Computing Machinery. — 2016. — SIGMOD '16. — P. 821–833. — Access mode: https://doi.org/10.1145/2882903.2915203.

[7] Repinskiy V. and Kovalenko V. DepMiner: A Pipelineable Tool for Mining of Intra-Project Dependencies. — 2021. — 2104.09473.

[8] Hulsebos Madelon, Hu Kevin, Bakker Michiel, Zgraggen Emanuel, Satyanarayan Arvind, Kraska Tim, Demiralp Çagatay, and Hidalgo César. Sherlock: A Deep Learning Approach to Semantic Data Type Detection // Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining. — New York, NY, USA : Association for Computing Machinery. — 2019. — KDD '19. — P. 1500–1508. — Access mode: https://doi.org/10.1145/3292500.3330993.

[9] Yao Hong, Hamilton Howard J., and Butz Cory J. FD_Mine: Discovering Functional Dependencies in a Database Using Equivalences // Proceedings of the 2002 IEEE International Conference on Data Mining. — USA : IEEE Computer Society. — 2002. — ICDM '02. — P. 729.