Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем

Группа 20.Б11-мм

Коробицын Игорь Андреевич

Расширения набора простых статистик в Desbordante статистиками для строковых данных

Отчёт по учебной практике
в форме «Решение»

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

Санкт-Петербург
2024

Оглавление

Введение

Профилирование данных — это процесс получения метаданных. Термином “метаданные” обозначаются любые данные о самих данных, т.е. любая информация, что-то говорящая об оригинальном массиве данных, будь то дата создания, количество столбцов, или же какие-то более подробные сведения о данных.

Профилирование данных может быть наукоёмким: тогда оно заключается в автоматизированном поиске в массиве данных неочевидных закономерностей, выраженных через функциональные зависимости или другие языки задания ограничений целостности, которые могут использованы, например, при постановке научных гипотез. Однако, более простые метаданные также могут быть полезны: например среднее арифметическое всех чисел в определённом столбце (если тип данных в нём численный) или максимальное количество слов в нём среди всех его ячеек (если тип данных в нём строковый). Такие метаданные далее будут называться *простыми статистиками*.

Desbordante [8] — это профилировщик данных, направленный как раз на наукоёмкое профилирование. Он разрабатывается на C++ с 2019-ого года и является open-source проектом с кодом на GitHub. Из-за фокуса на наукоёмком профилировании, он изначально несколько уступал аналогам в плане доступного функционала. Михаилом Фирсовым [10] и Павлом Аносовым [11] уже были реализованы многие из простых статистик, однако Desbordante всё ещё уступает аналогам в плане простых статистик, особенно для анализа строковых данных.

Целью данной работы является реализация таких статистик. Таким образом, эта работа является третьей работой по расширению набора простых статистик Desbordante.

1. Постановка задачи

К моменту начала работы Михаилом Фирсовым была проведена работа по обзору существующих профилировщиков данных и реализованных в них простых статистик, а также по реализации части этих статистик в Desbordante [10]. Эта работа была продолжена Павлом Аносовым, который также реализовал возможность их использования в Python с помощью библиотеки pybind11 [11].

Целью настоящей работы является дальнейшее расширение набора простых статистик в Desbordante, с добавлением возможности и их тоже вызывать из Python-кода. Для достижения этой цели были поставлены следующие задачи:

- Реализовать методы для вычисления ряда статистик для строковых столбцов в C++;
- Создать модульные тесты на C++, проверяющие их корректность;
- Сделать Python-привязки для статистик;
- Сделать тесты для проверки корректности работы статистик в Python;
- Провести экспериментальное нагрузочное тестирование и оценить производительность реализаций статистик.

2. Обзор

2.1. Глобальная архитектура

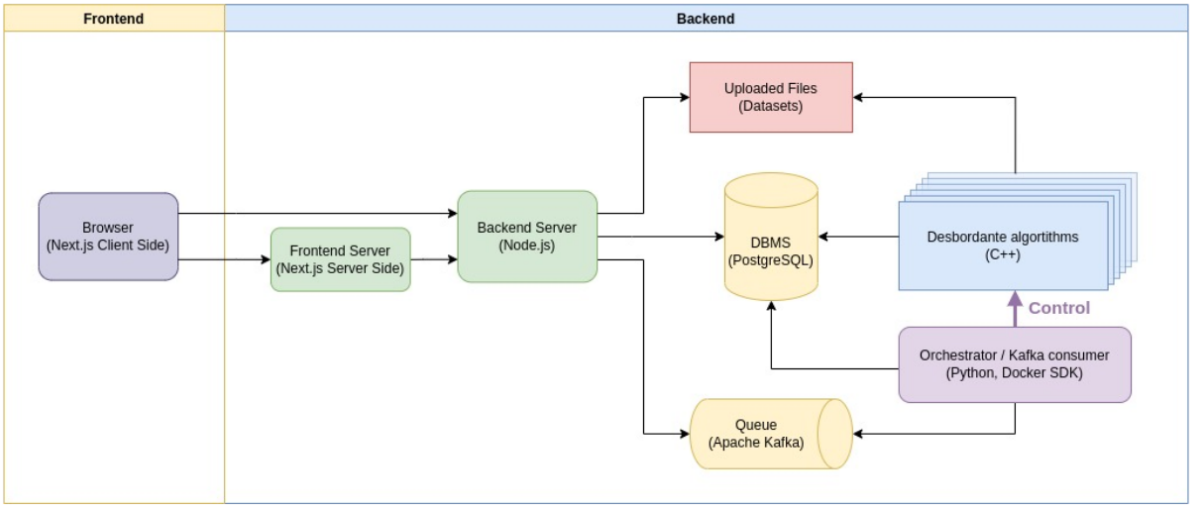

Рис. 1: Архитектура веб-приложения Desbordante

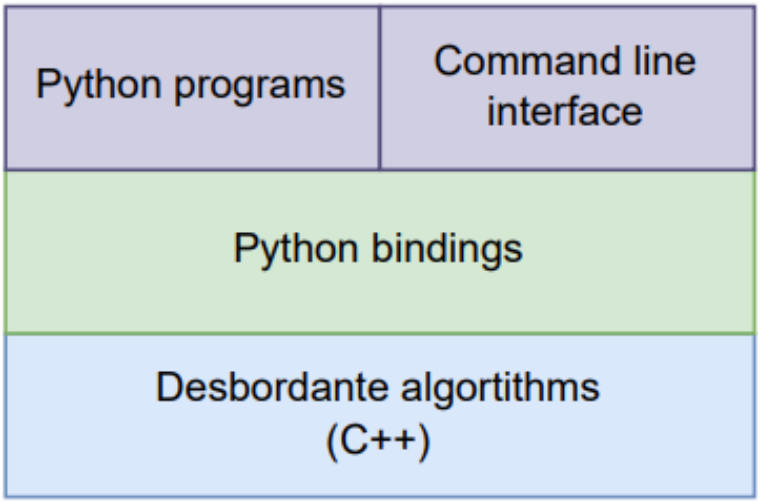

Рис. 2: Диаграмма ядра Desbordate, на которой видно 2 других интерфейса: Python библиотека и командная строка

Ядром архитектуры проекта является библиотека Desbordante (на рисунке 1, который отображает архитектуру веб-приложения, это блок Desbordante algorithms), в которой все алгоритмы реализованы на C++. Вся функциональность реализована через обращения к этой библиотеке через три интерфейса:

- Python библиотека: преимущество данного интерфейса заключается в том, что Python является одним из главных языков для работы с массивами данных. Во многом это связано с тем, что под него есть многие другие библиотеки, помимо Desbordante, для работы с данными (подробнее в разделе 2.2), благодаря чему можно одновременно использовать достаточно разнообразный функционал;

- Консоль: использование того же функционала, но через командную консоль;

- Веб-приложение: позволяет работать с массивами данных из браузера. Использует контейнеризацию и имеет микросервисную архитектуру, призванную распределять нагрузку. Как видно на рисунке 1, отдельные сервисы отвечают за:

  – Обработку пользовательских запросов;

  – Управление контейнерами (Orchestrator);

  – Работу с базами данных (DBMS);

  – Очередь выполнения задач (Queue).

2.2. Аналоги

Одним из аналогов — инструментом для анализа данных — является Pandas [1]. Он опирается на другую Python библиотеку: Numpy, которую сложно назвать аналогом так как она призвана работать с математическими объектами, а не данными. Анализ данных в Pandas осуществляется в основном через Pandas Profiling [2], который на основе массива данных генерирует отчёт. Он составляется весь разом, что может быть не самым эффективным решением если надо узнать только что-то конкретное.

Есть и другие Python библиотеки, которые используются для анализа данных: к таким относится AutoViz [3]. Его функционал в основном

ориентирован на визуализацию данных, а не под генерацию каких-то точных выводов программными методами.

Ещё один язык, часто используемый при работе с данными — это R. Для него есть, например, библиотека DataExplorer [4], в чём-то похожая на Pandas, и имеющая те же достоинства и недостатки.

Помимо библиотек, существуют и консольные приложения: например, Metanome [5, 6]. Главным его недостатком является то, что все алгоритмы написаны на Java, что сказывается на производительности. Из плюсов, в него можно относительно легко подключать новые алгоритмы.

Также стоит упомянуть Openclean [9] — ещё одно решение на Python. Оно базируется на других библиотеках для Python — например, тот же Pandas — и имеет несколько расширений, в числе которых графический интерфейс, который интегрирован с Jupyter, и расширение, позволяющее в Openclean использовать алгоритмы из Metanome.

Другой аналог — SPSS [7]. Это инструмент, разрабатываемый компанией IBM. Один из главных его недостатков заключается в том, что базовый функционал очень прост, а более сложный не доступен без наличия специальных знаний в области статистики.

Во многом SPSS является более профессиональным вариантом Excel или LibreOffice Calc, которые тоже можно отнести к инструментам для анализа данных. Как и в случае с SPSS, работа с ними ведётся через графический интерфейс, что отличает эти инструменты от Pandas, AutoViz и Metanome. Они не интегрируются с программным кодом, но зато более доступны не программистам.

Многие из приведённых примеров ориентированы скорее на работу с данными в более широком смысле. Тем не менее, функционал, который можно отнести к *профилированию* данных, в них присутствует тоже.

Итого, можно ввести следующую классификацию инструментов для профилирования данных, основанную на интерфейсе и применении:

- Текстовые решения:
  - Библиотеки для языков программирования: удобно исполь-

зовать совместно с другими инструментами и библиотеками того же языка (или даже другого: например, алгоритмы из Metanome (Java) в Openclean (Python));

- Консольные приложения: только алгоритмы и ничего лишнего. Удобно, если нужно, например, быстро что-то посмотреть, ничего не запуская и не компилируя;

- Графические решения:
  - Для не программистов: для пользователей, обладающих специальными знаниями в области статистики, но не в области программирования;
  - Для рядового пользователя.

Стоит также упомянуть, что многие из решений в первой категории тем не менее имеют некоторый графический интерфейс, что связано со спецификой предметной области: сложно представить себе анализ данных без возможности нарисовать график или посмотреть на выборку из таблицы. Например, Openclean и Pandas, являясь Python библиотеками, имеют графический интерфейс, реализованный через Jupyter Notebook. Речь идёт скорее об основном инструменте взаимодействия.

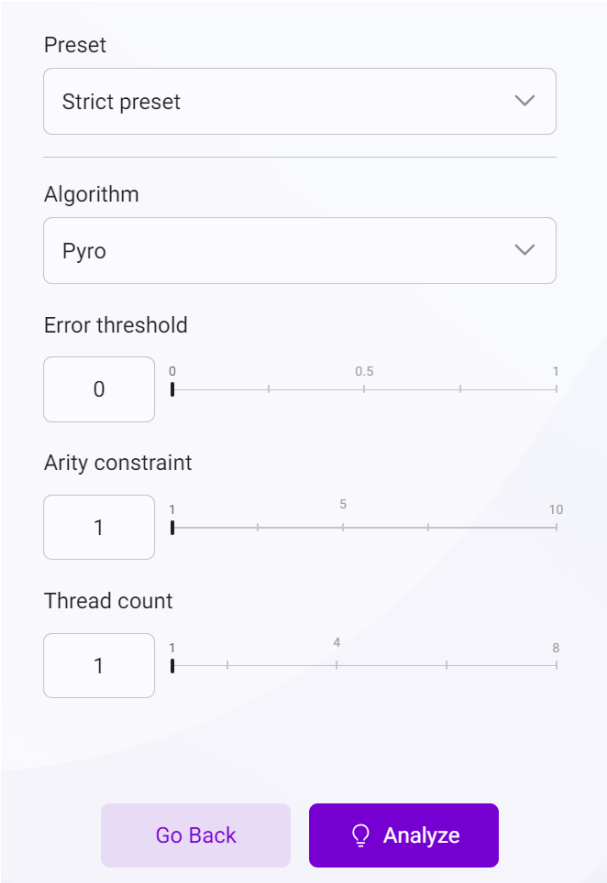

Рис. 4: Графический интерфейс веб-приложения Desbordante

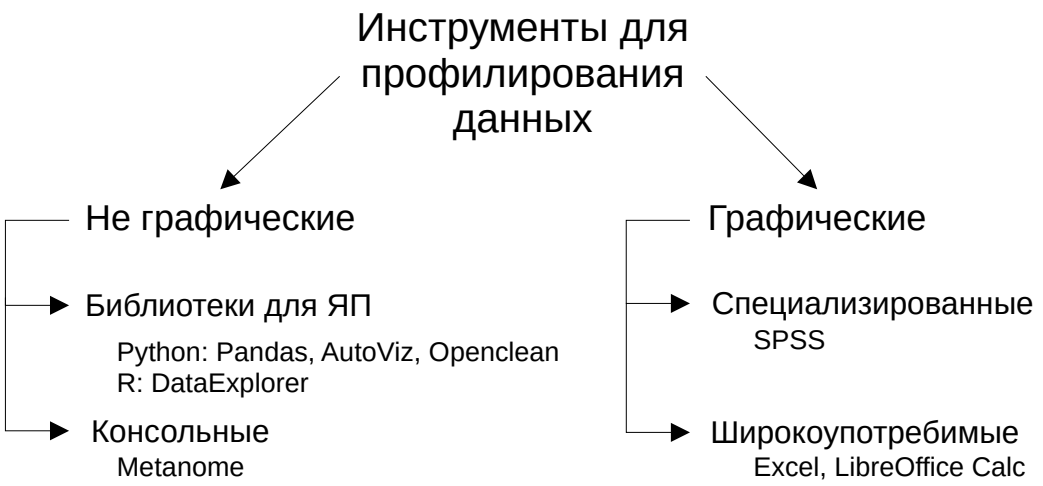

Рис. 3: Классификация и примеры

2.3. Локальная архитектура и детали устройства

Все методы, вычисляющие простые статистики, принадлежат классу DataStats. Помимо этих методов, в классе также присутствуют вспомогательные методы, и метод, разом вычисляющий все статистики, не требующие на вход никаких дополнительных параметров. Также, вычисленные значения для таких статистик сохраняются, чтобы не вычислять их каждый раз заново, если они понадобятся снова.

Практически все статистики берут на вход ровно одно значение: индекс столбца, на котором необходимо её подсчитать. Некоторые, впрочем, берут на вход дополнительные параметры: например, статистика top_k_chars, которая считает k наиболее частых символа в столбце, берёт на вход значение k. Возвращают же практически все статистики объект типа Statistic, который может хранить пустое значение и работает на собственных для Desbordante реализациях типов. Такие типы как set и vector не реализованы, поэтому если результатом работы статистики является значение такого типа, то метод, её вычисляющий, возвращает именно этот тип, а не Statistic. Примером такой статистики также может быть top_k_chars, возвращающий вектор из k элементов.

3. Реализация

В работе Михаила Фирсова [10] представлен список простых статистик, имеющихся в других профилировщиках данных. По итогам его работы, а также работы Павла Аносова [11], часть из них была реализована. Дальнейший выбор был обусловлен тем, что в большинстве своём эти статистики работают с числовыми данными, оставляя достаточно много нереализованных статистик для работы со строками.

3.1. Реализованные статистики

В ходе выполнения поставленной задачи были реализованы следующие статистики:

- words — все уникальные слова в столбце;
- top_k_chars — k наиболее частых символов в столбце;
- top_k_words — k наиболее частых слов в столбце;
- min_number_of_words — наименьшее количество слов среди ячеек столбца;
- max_number_of_words — наибольшее количество слов среди ячеек столбца;
- number_of_words — общее количество слов;
- min_number_of_chars — наименьшая длина строки среди ячеек столбца;
- max_number_of_chars — наибольшая длина строки среди ячеек столбца;
- number_of_entirely_lowercase_words — количество слов полностью в нижнем регистре;

- number_of_entirely_uppercase_words — количество слов полностью в верхнем регистре.

Также, с помощью библиотеки pybind11 была добавлена возможность использовать эти статистики в программах на Python.

3.2. Детали реализации

В ходе выполнения работы был написан код, описание которого приведено ниже. Он встраивался в класс DataStats. Почти все статистики возвращают в результате своей работы объект типа Statistic:

- Чтобы избежать переиспользования кода, был реализован ряд методов: GetStringMinOf, GetStringMaxOf и GetStringSumOf. Они возвращают минимум, максимум и сумму соответственно среди значений, вычисленных на каждой из ячеек строкового столбца с помощью некоторого предиката. Передавая в эти методы предикаты для подсчёта числа символов/слов соответственно, уже можно получить статистики:
  - min_number_of_words,
  - max_number_of_words,
  - number_of_words,
  - min_number_of_chars,
  - max_number_of_chars,

  а также избежать переиспользования кода в уже реализованной статистике number_of_chars;
- Две статистики были переделаны, и потому в pull request'е реализованы менее эффективно: number_of_entirely_lowercase_words и number_of_entirely_uppercase_words. Они проходят по каждой строке, и через булевскую переменную сохраняют, были ли символы в нижем или верхнем регистре. Эта переменная по умолчанию

имеет значение “истина”, и ей присваивается значение “ложь” если слово начинает не подходить. Когда слово заканчивается, значение переменной проверяется и снова меняется на “истина”. Таким образом, подсчёт осуществляется за один простой проход по строковой ячейке. Данное решение не является изначальной реализацией, неэффективность которой была выявлена в ходе экспериментов;

- Также, для этих статистик и статистик words и top_k_words был реализован метод, возвращающий список слов в строке;

- top_k_chars и top_k_words хранят словари, подсчитывающие для каждого уникального символа (или, соответственно, слова) сколько раз оно встречается. Потом они сортируются по числу использований, и из этого списка берутся первые k элементов;

- для работы со статистиками в Python использовалась библиотека pybind11: для всех статистик были сделаны Python-привязки, чтобы код методов, написанный на C++, можно было использовать, вызвав соответствующий метод в Python.

Простые статистики для работы со строковыми данными — как реализованные, так и уже имеющиеся — можно разделить на 3 класса по производительности и общему принципу работы. Эта классификация будет приведена в разделе с экспериментами.

3.3. Разработка модульных тестов

Для тестирования был выбран массив данных TestDataStats.csv, уже использующийся для тестирования простых статистик. Он был дополнен колонкой со строками, разбитыми на слова, чтобы протестировать статистики, работающие со словами и их подсчётом. Также были написаны тесты на Python, чтобы проверить, что Python-привязки статистик тоже работают корректно. Они тестировались на том же массиве

данных. Было сделано в общей сложности 10 тестов, по одному на каждую статистику.

4. Эксперимент

В эксперименте измеряется время вычисления не только добавленных строковых статистик, но и уже реализованных до этого Павлом Аносовым [11]. Описание добавленных статистик уже было приведено. Стоит также упомянуть уже имевшиеся строковые статистики:

- vocab — различные уникальные символы, встречающиеся в столбце;
- number_of_chars — общее количество символов в столбце;
- avg_number_of_chars — среднее количество символов среди ячеек столбца;
- number_of_uppercase_chars — общее количество букв в верхнем регистре;
- number_of_lowercase_chars — общее количество букв в нижем регистре;
- number_of_digit_chars — общее количество символов цифр;
- number_of_non_letter_chars — общее количество символов, не являющихся буквами.

Измерялось только время вычисления статистик, без времени, необходимого для начальной обработки тестовых данных.

4.1. Условия эксперимента

Эксперимент проводился на машине с процессором 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz и оперативной памятью 16 ГБ в виртуальной среде Ubuntu (64 bit) на операционной системе Windows 11. Сама виртуальная среда была создана в Oracle VM Virtual Box, и ограничена 4 ГБ оперативной памяти и 2 потоками процессора.

4.2. Исходные данные

Эксперимент проводился на массиве данных iowa1kk.csv, в котором помимо строковых присутствуют также и числовые столбцы, что позволяет протестировать помимо строковых числовые статистики тоже. В нём 1 000 000 записей и 24 столбца, 8 из которых — строковые. Он имеет размер 214 мегабайт.

4.3. Исследовательские вопросы

Для того, чтобы оценить качество предлагаемого решения, было необходимо понять насколько быстро вычисляются новые статистки, а также насколько алгоритмы для их вычисления масштабируемы. Для этого были поставлены следующие исследовательские вопросы:

- RQ1: Сколько времени занимает вычисление добавленных статистик?
- RQ2: Как это время отличается от статистики к статистике и как сравнимо со временем вычисления уже имеющихся строковых статистик?
- RQ3: Какова зависимость времени вычисления статистик от объёма данных?

4.4. Эксперимент

Для того, чтобы ответить на исследовательские вопросы, тестирование было проведено следующим образом:

- На основе набора данных iowa1kk.csv были созданы отдельные наборы данных, содержащие первые 100 тысяч его записей, первые 200 тысяч, и так далее до одного миллиона;
- Для каждого из этих наборов данных нагрузочное тестирование было проведено 10 раз на всех строковых статистиках по отдельности, а также на всех статистиках вместе (включая нестроковые);

- Для каждой строковой статистики, а также для всех статистик вместе, для всех вариантов количества записей было взято среднее арифметическое времени работы этой статистики на этом наборе данных;
- Эти данные были внесены в таблицу и на их основе построены графики.

Ниже приведено время работы для всех строковых статистик, поддерживаемых Desbordante, на массиве данных iowa1kk.csv по данным этого эксперимента (среднее по 10 запускам на полном массиве данных):

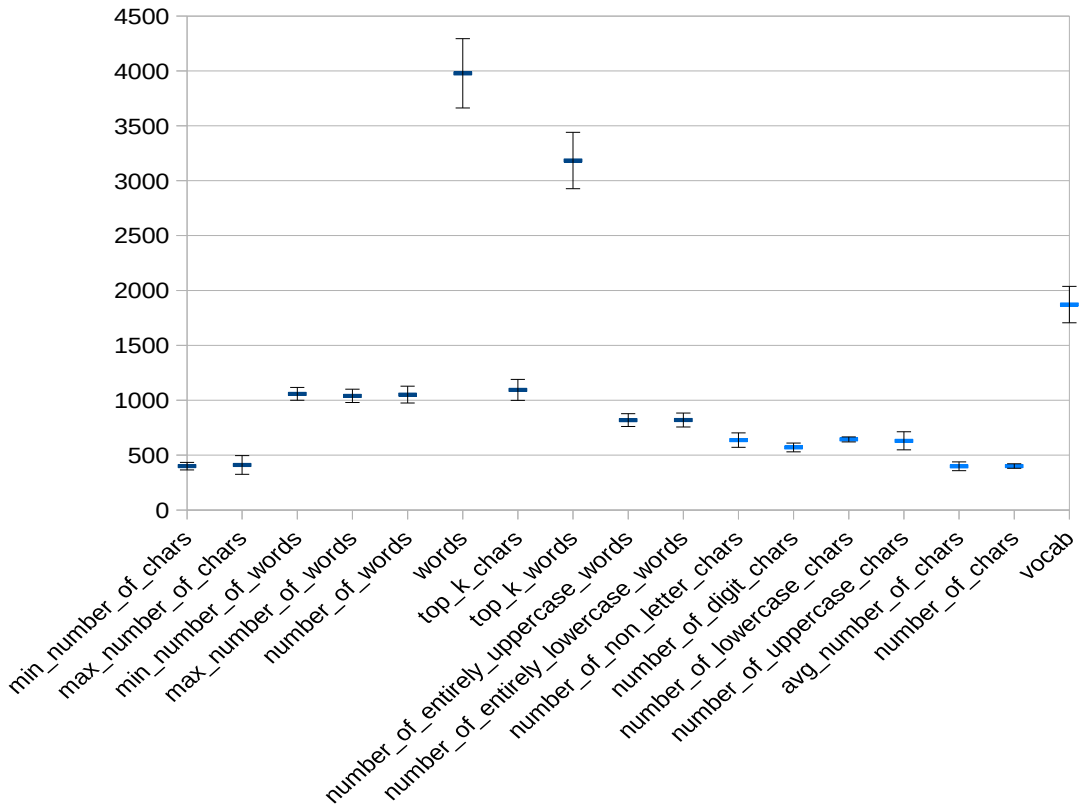

Рис. 5: Время работы строковых статистик в миллисекундах (более светлым оттенком синего обозначены уже имевшиеся статистики)

На графике на рисунке 5 показано время вычисления каждой стати-

стики на массиве данных с 1 миллионом записей. Их можно разделить на несколько категорий:

- Простой подсчёт всех символов: number_of_chars, а также среднее, минимальное и максимальное их количество. Они работают быстрее всех;
- Подсчёт символов, удовлетворяющих определённому условию: в верхнем или нижем регистре, или только цифр или только символов, не являющихся буквами. Они работают чуть дольше, чем предыдущая группа. Сюда же можно отнести статистики, подсчитывающие слова: по сути, их количество — это количество непробельных символов, следующих сразу после пробела или начала строки (или сразу перед пробелом или концом строки);
- Списковые: vocab, top_k_words, top_k_chars, words. Они возвращают отсортированный вектор или множество (или отсортированную строку в случае vocab). Они работают дольше других ввиду времени, которое требуется, чтобы составить ответ.

Рис. 6: Пример роста времени работы статистики в зависимости от размера массива данных

Зависимость времени работы от числа записей линейная для всех статистик. Пример можно увидеть на графике на рисунке 6. Та же зависимость соблюдается для всех строковых статистик, в том числе и тех, которые уже были до этого.

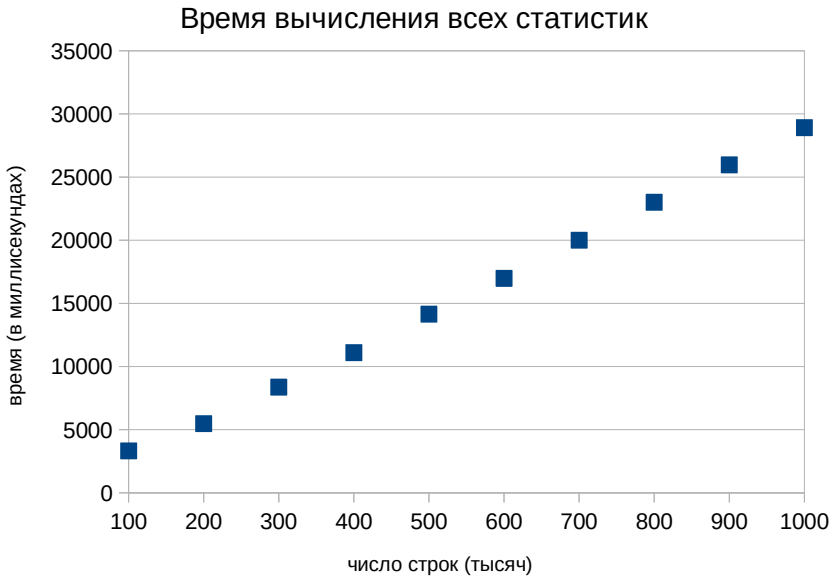

Рис. 7: Результаты эксперимента с подсчётом всех статистик

На графике на рисунке 7 видно, что при одновременном подсчёте всех простых статистик в Desbordante также выполняется линейная зависимость.

4.5. Выводы

Если подвести итоги нагрузочного тестирования, то можно сделать следующие выводы:

- RQ1: на массиве данных iowa1kk.csv с одним миллионом записей и восемью строковыми столбцами вычисление каждой из строковых статистик занимает где-то от 500 до 4000 миллисекунд, а вычисление вообще всех простых статистик Desbordante — примерно 30 секунд;

- RQ2: статистики можно разделить на три категории. Быстрее всего вычисляются статистики, просто подсчитывающие все символы без разбора: менее 500 миллисекунд. Далее идут статистики, подсчитывающие слова или символы, удовлетворяющие какому-то условию: от 500 до 1000 миллисекунд. Дольше всего вычисляются статистики, результат работы которых это строка, множество или вектор: top_k_chars (чуть больше 1000 миллисекунд), vocab (около 2000 миллисекунд), top_k_words (более 3000 миллисекунд) и words (4 секунды);
- RQ3: зависимость времени от числа записей линейная для всех статистик.

5. Итоговая таблица статистик

В таблице ниже приведены статистики, которые поддерживаются в Desbordante на момент окончания производственной практики. Более подробное описание статистик можно найти в работе Михаила Фирсова [10].

Таблица 1: Поддерживаемые статистики

Тип статистики	Статистика	есть в C++	есть в питоне
String	vocab	+	+
	words	+	+
	topKChars	+	+
	topKWords	+	+
	minWords	+	+
	maxWords	+	+
	wordCount	+	+
	nonLetterChars	+	+
	diacriticChars	-	-
	digitChars	+	+
	lowercaseChars	+	+
	uppercaseChars	+	+
	ExclFirstLetters	-	-
	minWhiteSpaces	-	-
	maxWhiteSpaces	-	-
	avgChars	+	+
	minChars	+	+
	maxChars	+	+
	totalCharCount	+	+
	entirelyLowercaseCount	+	+
	entirelyUppercaseCount	+	+
General	dataType	+	+
	columnName	+	+

	categorical	+	+
	samples	+	+
	min	+	+
	max	+	+
	quantiles	+	+
	nullCount	+	+
	uniqueCount	+	+
	sampleSize	+	+
	categoricalCount	-	-
	uniqueRatio	-	-
	categories	+	-
Float	precision	-	-
	sampleRatio	-	-
DateTime	highestTime	-	-
	lowestTime	-	-
Numeric	sum	+	+
	mean	+	+
	median	+	+
	geometricMean	+	+
	variance	+	+
	correctedSTD	+	+
	centralMoment	+	+
	standardizedCentralMoment	+	+
	skewness	+	+
	kurtosis	+	+
	meanAbsoluteDeviation	+	+
	medianAbsoluteDeviation	+	+
	numZeros	+	+
	numNegatives	+	+
	biasCorrection	-	-
	histogram	-	-
	historgramAndQuantiles	-	-

	sumOfSquares	+	+
Bool	trueCount	-	-
	falseCount	-	-
Tableau	columnCount	+	+
	rowHasNullRatio	+	+
	rowIsNullRatio	+	+
	uniqueRowRatio	+	+
	duplicateRowCount	-	-
	fileType	-	-
	encoding	-	-
	correctionMatrix	-	-
	chi2Matrix	-	-
	profileSchema	-	-

Заключение

В процессе работы были достигнуты следующие результаты:

- В C++ реализованы методы для получения ряда статистик для строковых столбцов;
- Были созданы модульные тесты на C++ для проверки корректности работы методов;
- Сделаны python-привязки статистик в Python;
- Тесты на C++ продублированы на Python для проверки корректности работы привязок;
- Проведено экспериментальное нагрузочное тестирование и оценена производительность реализаций статистик.

Принят pull request, код доступен в репозитории Desbordante[1].

[1] https://github.com/Desbordante/desbordante-core/pull/403

Список литературы

[1] URL: https://pandas.pydata.org/docs/ (online; accessed: 2024-08-30).

[2] URL: https://pypi.org/project/pandas-profiling/ (online; accessed: 2024-08-30).

[3] URL: https://readthedocs.org/projects/autoviz/ (online; accessed: 2024-08-30).

[4] URL: https://cran.r-project.org/web/packages/DataExplorer/vignettes/dataexplorer-intro.html (online; accessed: 2024-08-30).

[5] URL: https://hpi.de/naumann/projects/data-profiling-and-analytics/metanome-data-profiling.html (online; accessed: 2024-08-30).

[6] URL: https://github.com/HPI-Information-Systems/Metanome (online; accessed: 2024-08-30).

[7] URL: https://www.ibm.com/support/pages/ibm-spss-statistics-28-documentation (online; accessed: 2024-08-30).

[8] Desbordante: a Framework for Exploring Limits of Dependency Discovery Algorithms / Maxim Strutovskiy, Nikita Bobrov, Kirill Smirnov, George Chernishev. — 2021. — URL: https://ieeexplore.ieee.org/document/9435469 (online; accessed: 2024-05-06).

[9] From Papers to Practice: The openclean Open-Source Data Cleaning Library / Heiko Müller, Sonia Castelo, Munaf Qazi, Juliana Freire. — URL: https://vldb.org/pvldb/vol14/p2763-mueller.pdf (online; accessed: 2024-05-06).

[10] Михаил Фирсов. Реализация статистик Desbordante. — 2023. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/Statistics%20-%20Mikhail%20Firsov%20-%202022%20autumn.pdf (online; accessed: 2024-05-06).

[11] Павел Аносов. Расширение набора простых статистик в "Desbordante". — 2024. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/Statistics%20-%20Anosov%20Pavel%20-%202023%20autumn.pdf (online; accessed: 2024-05-06).