Автоматический контроль качества конвейеров данных

Королев А.С., СПбГУ, Санкт-Петербург st087375@student.spbu.ru,
Благов М.В., СПбГУ, Санкт-Петербург m.blagov@spbu.ru

Аннотация

В рамках данной статьи было разработано приложение для автоматического контроля качества данных, состоящее из двух модулей – генератора данных и программы для проверки конвейера. Приложение реализовано на языке программирования Python с использованием библиотек Faker, Pandas, Pyspark. Приложение валидирует запрашиваемые данные, использует разработанный алгоритм генерации датасетов и сравнивает результаты своей работы с работой конвейера даннных. Приложение позволяет тестировать конвейеры данных, которые работают на основе операций: left join, right join, outter join, inner join.

Введение

Конвейер данных – приложение, используемое для передачи данных. Большинство современных IT компаний используют конвейеры данных для доставки данных в аналитическое хранилище с целью использования этой информации для улучшения своих продуктов и сервисов[4]. Как и любое другое приложение конвейер данных необходимо тестировать, но тестирование конвейеров данных отличается от тестирования других приложений. Конвейеры данных представляют собой интеграционные пайплайны, на входе и на выходе у которых датасеты. Из чего вытекают следующие проблемы:

- Необходимо тестировать входные данные
- Разные части конвейера данных реализованы при помощи разных инструментов и все нужно протестировать
- Трудоемкость подготовки тестовых данных

Среди перечисленных проблем первые две уже имеют решения и существуют специальные профессии, в сферу обязанностей которых входит решение данных трудностей[1].

Последняя проблема на текущий момент не имеет готового решения и является ключевой для данной работы[5]. Дополнительно отметим, что для тестирования нельзя использовать данные из продуктовой среды из-за политики обработки чувствительных данных.

Постановка задачи

Конвейер данных

Математически конвейер данных может быть описан как отображение T, определенное на множестве входных значений в множество выходных значений. В нашем случае множество входных значений представляет собой кортеж датасетов $D_1, ..., D_n$, а множество выходных значений - датасет D'. Датасетом мы будем называть набор колонок и строк $D_i = (r_k, c_j)_{k,j=1}^{n}$, где каждая колонка может являться одним из следующих типов данных: целое значение, значение с плавающей точкой, дата, булевское значение, строки произвольной длины.

$$T : D_1 \times D_2 \times D_3 \times ... \times D_n \mapsto D' \tag{1}$$

В течение работы конвейера над изначальными датасетами производится набор операций-преобразований. Обозначим за T_i преобразование на i-том этапе работы конвейера данных.

$$T_i : (D_1, D_2, ..., D_n) \mapsto D' \tag{2}$$

После i-го этапа выходные данные передаются на следующий этап и конвейер работает с этими данными как с входными. Если обозначить считывание данных как R, а запись как W, которые можно определить как:

$$R : D_1 \times D_2 \times D_3 \times ... \times D_n \mapsto (D_1, D_2, ..., D_n) \tag{3}$$

$$W : (D'_1, D'_2, ..., D'_n) \mapsto D' \tag{4}$$

Благодаря введеным определениям. конвейер данных можно представить как композицию отображений:

$$T(D_1 \times D_2 \times D_3 \times ... \times D_n) = W(T_n(T_{n-1}(...(T_1(R(D_1 \times D_2 \times D_3 \times ... \times D_n)))))) \tag{5}$$

Также для итоговых датасетов введем определение эквивалентности 2 датасетов.

Определение: 2 датасета эквивалентны ($\sim$), если датасеты содержат одинаковые записи с точностью до перестановки строк и столбцов.

Генерация датасетов

Чтобы протестировать конвейер данных, нам необходимо заранее будет подготовить датасеты. Для подготовки n датасетов мы требуем на вход n соотвествующих схем, $n-1$ связь между датасетами, $n-1$ условие соединения и последовательность размером n ключей, которые должны быть одинаковы в следующих друг за другом датасетах.

Таким образом задача тестирования конвейеров данных формулируется так: необходимо сгенерировать такой набор датасетов $D_1, ..., D_n, D'$, чтобы при известном T выполнялось равенство:

$$T(D_1, ..., D_n) = D' \tag{6}$$

Поставив задачу, перейдем к алгоритму генерации датасетов, который позволит подготовить тестовые данные для конвейера данных.

Генерация удовлетворительных данных

Алгоритм генерации датасетов: генерация произвольного D_i

Прежде всего договоримся, что на вход в нашу программу будет подаваться конфигурационный файл с описанием датасетов. Прежде чем описать требования к конфигурационному файлу, введем определение схемы.
Определение: Схемой датасета называется список столбцов этого набора данных с описанными типами данных и их ограничениями.
От конфигурационного файла мы будем требовать:

1. Количичество датасетов и соответствующие им схемы
2. Размеры датасетов и условиях их соединения
3. Количество коррелирующих ключей в датасетах
4. Ограничения для каждого столбца в датасете или список возможных значений для конкретной колонки в датасете

Конфигурационный файл будем называть корректным, если выполняюся следующее

1. Количество датасетов равно количеству коррелирующих ключей - 1 и равно количеству соединений -1

2. Для соединения используются только операции inner-join, outer-join,right-join,left-join

3. Последовательность коррелирующих ключей является невозрастающий

4. Ограничения на столбцы позволяют сгененерировать столько значений, сколько требуется размерами датасета

В дальнейшем будем отталкиваться от того, что пользователь ввел корректную конфигурацию. В качестве T_i рассматриваем соединение по одному или нескольким полям.

Алгоритм генерации датасетов: генерация D_1, D_2 (база)

Сперва возьмем 2 схемы для датасетов A_1, A_2 с размерами n, m соответственно. Затем сперва сгенерируем k_1 ключей соединения, которые будут не совпадать между собой, но будут присутствовать в обоих датасетах:

$$\forall i \neq j, i \in \{1 : k_1\}, j \in \{1 : k_1\} : x_i \neq x_j, \forall x_i, x_j \in \{x_1, x_2, ..., x_{k_1}\} \quad (7)$$

Затем генерируем оставшиеся $n - k_1, m - k_1$ ключей так, чтобы они не пересекались между собой и не совпадали с исходными k_1 ключами соединения. После того, как все ключи соединения были сгенерированы, генерируем данные для оставшихся столбцов в обоих датасетах и выполняем операцию соединения, заданную пользователем и получаем датасет $A_{12} = A_1 \; join \; A_2$, длина которого равняется k_1 в случае inner-join.

Алгоритм генерации датасетов: генерация D_n, D_{n+1}(переход)

Теперь нам необходимо сгенерировать датасет A_3 размером l с k_2 ключами соединений, имея на руках датасет A_{12}. Для того, чтобы это сделать скопируем первые k_2 значений из датасета A_{12} и вставим в соответствующие столбцы. Для столбцов, которые отстутствовали в таблице A_{12} будем генерировать новые значения.

После копирования генерируем оставшиеся $l - k_2$ ключей соединения так, чтобы они не пересекались с исходными k_2 ключами и между собой. Затем также генерируем оставшиеся значения для датасета A_3 и объеденяем его с датасетом A_{12} по соединению, которое задал пользователь и получаем датасет $A_{123} = A_{12} \; join \; A_3$

Алгоритм генерации датасетов: результат

После выполнения приложения у нас будут датасеты: $A_1, ..., A_n$, сохраненные в формате CSV или Parquet (в зависимости от выбора польователя) и итоговый датасет $A_{123...n}$

Реализация

Входные данные

На вход в программу пользователь передает yaml-файл с конфигурацией для генерации датасетов, путь до конвейера данных, путь до получившихся датасетов и путь для сохранения результата для spark приложения. На выходе пользователь получает ответ относительно результата работы конвейера: получили ожидаемый результат или неожидаемый и надо исправлять код конвейера. Пример конфигурационного файла[2]

Структура приложения

Приложение состоит из 3 классов: config_manager,sample_generator,tester.

Класс config_manager

Класс config_manager парсит схему и валидирует ее. При валидации запускаются набор тестов, которые проверяют ее на валидность, а именно:

- Проверка на количество датасетов, соединений и количество коррелирующих ключей и размеры датасетов. Между ними должно быть следующее соответствие: количество схем датасетов = количество соединений + 1, количество схем датасетов = количество коррелирующих ключей + 1, а также последовательность коррелирующих ключей должна быть не строго убывающей, а также размеры датасетов должны быть не меньше количества требуемых коррелирующих значений.

- Проверка на то, что общие ограничения, уникальные ограничения и списки возможных значений соответствуют размерам датасета.

- Проверка на то, что коррелирующие поля обладают одним типом данных

- Проверка на то, что при всевозможных различных ограничениях на коррелирующие столбцы количество возможных ключей удовлетворяет размерам датасета

В случае, если хотя бы 1 тест не прошел, поднимается ошибка с сообщением о необходимости исправить исходный конфигурационный файл.

Класс sample_generator

Этот класс реализует алгоритм генерации датасетов, который описан выше, а также агрегирует при заданном условии в конфигурации датасеты и сохраняет их в формате csv или parquet в указанной директории.

Класс tester

Класс tester - класс-тестировщик, который на вход принимает путь до конфигурационного файла, пути, где будут лежать итоговые датасеты для конвейера данных, путь до конвейра данных и путь, куда сохранить результат работы конвейера.

Сперва класс вызывает класс $config\_manager$, который парсит конфигурационный файл и валидирует его, а затем передает все данные в класс $sample\_generator$ и происходит генерация датасетов. После окончания генерации, запускается конвейер данных с сгенерированными датасетами, а затем берутся результаты работы генератора датасетов и конвейера данных и сравниваются с учетом введеного отношения эквивалентности. Если конвейер данных выдает неожидаемый результат, то поднимается ошибка, иначе выводится сообщение об успешной работе конвейера данных.

Полный код приложения можно найти в [3]

Заключение

В рамках данной работы было проведено исследование существующих инструментов и ничего не было найдено на тему автоматического контроля качества конвейеров данных. Был разработан и реализован алгоритм генерации датасетов, на основе которого было разработано CLI-приложение на языке программирования Python с использованием библиотек Faker, Pandas, PySpark для автоматического контроля качества конвейеров данных.

Список литературы

[1] Кто такой и чем занимается Data QA Engineer, 2021. URL: https://habr.com/ru/companies/skillfactory/articles/591061/

[2] Пример конфигурационного yaml-файла, 2024 URL: https://github.com/prostotema1/Graduation-work/blob/master/config.yaml

[3] Код приложения, 2024. URL: https://github.com/prostotema1/Graduation-work

[4] Что такое конвейер данных? И почему вы должны это знать, 2023. URL: https://habr.com/ru/sandbox/188820/

[5] Automating Data Quality Check in Data Pipelines,2023. URL:https://dzone.com/articles/automating-data-quality-check-in-data-pipelines

[6] Faker documentation,2024. URL:https://faker.readthedocs.io/en/master/

[7] PySpark documentation,2024.
URL: https://spark.apache.org/docs/latest/api/python/index.html