Путьков Дмитрий Александрович

Заславский Марк Маркович

Каф. МО ЭВМ, Программная инженерия,

15 декабря, 2017г.

Сбор статистики с репозиториев Github

Аннотация

В данной статье описаны исследования возможных решений для сбора информации с репозиториев по заданным критериям и обработка и сохранение данных результатов в гугл таблице. Преимуществом подхода, описанного в данной статье, является покрытие всех поставленных требований к инструменту подобного рода. Реализовано данное приложение на языке программирования Python, некоторые компоненты реализованы на GoogleAppScript(язык на базе JavaScript). Также в исследовании описываются некоторые аналоги выбранного инструмента, описаны их сильные и слабые стороны, обосновано, почему они не были выбраны в качестве решения поставленной задачи.

Введение

Причиной изучения данной темы стала необходимость собирать статистику с репозиториев, в которых располагаются проекты: количество pull requests , commits , issues.

Цель исследования - создание инструмента, позволяющего иметь визуальное представление об изменении статистики репозиториев с течением времени, сравнения этих репозиториев и их участников между собой.
Инструмент подобного рода - один из инструментов, которые необходимы на проектах с большим количеством людей, для отображения активности членов проектов. Реальный пример : Просмотр преподавателям активности студентов благодаря этому скрипту и сравнения результатов групп между собой. Также актуальность данного исследования подкрепляется относительно небольшим и, в большинстве своем, узконаправленными инструментами для сбора подобного рода статистики.

Задачами, которые должно было покрывать решение, стали :

- Сравнение несколько репозиториев между собой.
- Сравнение участников всех репозиториев между собой

- Визуализация представления статистики в виде графиков
- Графический показ изменения статистики с течением времени.

Обзор предметной области

Таким образом, объектом исследования является сбор статистики с репозиториев платформы Github,а именно поиск оптимального инструмента для нахождения всех интересующих критериев с репозитория, сравнения репозиториев между собой по критериям и представление результата в виде графиков.

Для начала были рассмотрены уже существующие способы сбора статистики, проведено сравнения их между собой по выделенными нами критериями:

сбор статистики в Github

Github имеет свой собственный инструмент для сбора статистики с репозитория широкому спектру критериев. Также он автоматически строит по этим критериям графики.

Стандартные свойства Git

У Git есть команда git shortlog –s -- author=author@gmail.com , которая получает количество коммитов по данному пользователю. Также можно получить более конкретную информацию(а именно : сколько строчек добавлено\удалено).

Caelum-git- reports

Это генератор статистики для репозиториев. Он может получать количество коммитов по всему репозиторию и по отдельному человеку, при этом учитывая промежуток времени, за который нам это интересно.

"Ручной" сбор информации

Наверно, самый распространенный, самый полный и самый долгий способ сбора статистики: Просматривать информацию индивидуально по каждому репозиторию и фиксируя все данные, которые тебя интересуют.

Критерии для сравнения:

Скорость получения данных

Этот критерий определяет, сколько понадобится времени для получения интересующих данных.

Ассортимент получаемых данных.

Данный критерий определяет, какие именно данные (коммиты, ишью или пулреквесты) могут быть получены при использовании одного из способов.

Удобство чтения данных

Этот критерий оценивает, насколько удобно будет воспринимать полученную информацию.

Удобство сравнения статистик разных репозиториев

Определяет, насколько удобно использовать способ для сравнения большого числа разных репозиториев и/или участников.

Таблица

Аналоги	Скорость	Ассортимент	Удобство для чтения	Удобство сравнения
Инструмент Github	Очень быстро/автоматически	Широкий	Имеются графики	Неудобно
Caelum	Быстро	Только коммиты	Без графиков	Неудобно
Ручной способ	Медленно	Широкий	Без графиков	Неудобно

Таким образом, можно сделать вывод, что некоторые из представленных в таблице способ эффективны в большинстве случаев, однако ни один из них не в состоянии выполнить все поставленные задачи полностью. Самым удобным можно выделить базовый инструмент GitHub, единственный минус которого в том, что графики строятся только по одному репозиторию. В остальном же, и количество информации, и широта рассматриваемых критериев полностью (и даже больше) покрывает нужды. Что касается остальных критериев, то они либо узконаправленные, либо слишком медленные (ручной способ). Таким образом, можно сделать вывод, что

существует достаточное количество аналогов, но ни один из них полностью не подходят для достижения поставленной цели. Именно поэтому было решено *создать* инструмент, который бы показывал хорошие результаты по всем данным критериям.

Выбор метода решения

На основании сравнения аналогов было решено, что:

- Решение должно представлять собой инструмент, который бы требовал минимальных затрат времени пользователя.
- Этот инструмент должен быть удобен в использовании, то есть следует свести действия пользователя к минимуму. Отличным решением являлся бы встроенный сбор статистики в Github, если бы он мог сравнивать репозитории между собой.
- Инструмент должен содержать в себе и возможность сравнения репозиториев и их участников между собой, а также сохранять статистику за время первого использования скрипта, для визуального представления развития репозитория.
- Кроме этого, было решено внедрить такое свойство, как построение графиков по этим репозиториям для более удобного просмотра статистики.
- Далее, следовало учесть возможность того, что некоторые участники проектов не важны для статистики (Пример: преподаватели в репозитории групп). Поэтому инструмент должен был быть снабжён возможностью игнорировать статистику указанных пользователей.
- Также следовало учесть, что может появится необходимость сравнения участников из разных репозиториев между собой, поэтому должны быть построены отдельные графики только по людям.

Описание метода решения

Архитектура

Архитектуру программной реализации в общем виде можно представить следующим образом:

1 Подключение к гугл-таблице (функция authorize_to_spreadsheets)
Для доступа к гугл-таблице требуется документ в формате .json, генерируемый прямо при создании проекта Google.
Затем, происходит подключение к таблице, передавая путь к этому файлу и ссылку https://spreadsheets.google.com/feeds

2 Генерация страниц и запись текущей даты (функция get_date)
Теперь, имея доступ к таблице, следует создать список листов, чтобы в дальнейшем записывать добавлять туда информацию. В переменную записывается текущая дата в формате дд-мм-гг

3 Получение и парсинг аргументов (функция get_credentials)
Используя библиотеку argparse, функция считывает следующие аргументы:

- CREDENTIALS название файла, хранящие client_id client_secret для Github
- REPO список репозиториев, информацию о которых требуется получить.
- IVISIBILITY_ACCESS файл, содержащий список участников, информацию о которых не требуется учитывать.
- LINK Ссылка на гугл-таблицу, в которую будет записываться информация

4 Учёт исключений, переданных как аргументы (функция check_invisibility)
создание списка участников, полученный через INVISIBILITY.

5 Запись данных в таблицу (функция creating_worksheet)
   i Авторизация на Github (функция authed_login)
   ii Проверка, создаем ли мы новую таблицу или обновляем уже существующую(функция whether_sheet_is_new)
   iii Создаем оформление : на одном листе будет 4 таблицы, каждый содержит название, столбцы(дату обновления), строки(репозитории или учатники).(функция create_head_for_general или create_head_for_users)
   iv Для каждого репозитория из списка вызывается функция, общая сложность которой O(countOfAssignees)+O(countOfIssues) +O(countOfCommits)+O(countOfPull() (функция Get_info)
   v Запись полученных данных в таблицы. (push_spread_for_generals или push_spread_for_users)

6 Данный пункт является самым долгим по времени выполнения. Это связано, прежде всего, с подключением к Github data и записи их в Google spreadsheet)

7 Скрипт, написанный на GoogleAPPScript, создающий графики
   - При первой записи данных в новую таблицу следует запустить скрипт, передав в него ссылку на таблицу. В дальнейшем это не требуется, т.к. скрипт создает графики по изначально очень большому рэнджу из клеток, поэтому графики будут изменяться самостоятельно.

Входные данные:
- Ссылка на таблицу
- файл с client_id, client_secret github
- файл со списком репозиториев
- файл со списком людей-исключений(необязательный параметр)

Выходные данные:
- Ссылка на созданную таблицу
- Ссылка на гугл-скрипт

Сценарий использования:
./get_info.py --credentials credFile --repos reposFile --invisibility peopleFile --link link

Используемые технологии:
- Библиотека PyGitHub для Python (2 and 3) позволяющая получать доступ к GitHub API v3. Эта библиотека позволяет управлять

ресурсами GitHub: repositories, user profiles, and organizations в приложении, написанном на Python.

- Модуль argparse упрощает создание удобных интерфейсов командной строки. Программа определяет, какие аргументы она требует, а arg-parse будет определять, как разбирать их из sys.argv. Модуль argparse также автоматически генерирует сообщения справки и использования и выдает ошибки, когда пользователи дают программе недопустимые аргументы.

- Библиотека gspread позволяет управлять таблицами Google.

Особенности:

Запись в таблицу по названию или ссылке.
oauth2client.service_account Это клиентская библиотека для доступа к ресурсам, защищенным OAuth 2.0.

• datetime Модуль datetime предоставляет классы для управления датами и временем как простым, так и сложным способом. Хотя арифметика даты и времени поддерживается, фокус реализации заключается в эффективном извлечении атрибутов для форматирования и обработки вывода. Для связанных функций см. Также модули времени и календаря.

Интерфейсом пользователя является командная строка.
Данные для участников хранятся в словаре, которая создается при итерации по списку репозиториев. Она обнуляется каждый раз при записи данных нового репозитория. Также существует общий список данных, в который заносится имя репозитория и информация по нему. Такой словарь имеет жизненный цикл, равный жизненному циклу работы программы.

Тесты, используемые при проверки работы :

- Интеграционный тест

Заключение

Таким образом, был создан инструмент, который оптимально подходит для решения поставленных задач. Скорость сбора информации с Github невысокая, но объем полученной информации высокий. Также полученная информация отображается в виде графиков- такая визуализация информации является удобной для восприятия. Можно выделить следующие недостатки данного выбора решения :

1. Необходимость самим создавать инструмент
2. Невысокая скорость сбора информации с репозиториев
3. Привязка к гугл-таблице и гугл-скриптам. В этом случае необходимо хранить файл-ключ в папке со скриптом для корректной работы приложения.
4. Не автоматизированная работа гугл-скрипта: при первом сборе информации по репозиторию, требуется вручную запустить гугл-скрипт для создания графиков. В дальнейшем, при обновлении информации по репозиторию, этого делать не требуется.

В итоге, цель, ради которой создавалось приложение, была достигнута. Были решены все поставленные задачи. Однако, в дальнейшем данный инструмент может быть доработан в следующих направлениях : полная автоматизация работы и внедрение графического интерфейса для более комфортной работы с приложением.

Список литературы

1. github.com/caelum/git-reports/wiki

2.https://pypi.python.org/pypi/argparse

3.https://www.twilio.com/blog/2017/02/an-easy-way-to-read-and-write-to-a-google-spreadsheet-in-python.html

4.https://github.com/PyGithub/PyGithub