Aufgabe 4 - Jascha Knack - 1270412
https://github.com/jaschaknack/fst_ws1617_exercises/tree/master/exercise_04

Teil 1 - Datenbasis zur Beantwortung der Forschungsfragen festlegen

Forschungsfragen:

- 2.1. Wie ändert sich die Frequenz der Commits mit Einführung von TravisCI?
- 2.2. Wie ändert sich die Größe der Commits (LOC oder Anzahl Dateien) mit Einführung von TravisCI?

Um die Forschungsfrage zu beantworten, wird zunächst der Zeitpunkt benötigt, an dem das Projekt mit TravisCI begonnen und ggf. wieder aufgehört hat. Des Weiteren wird für die erste Forschungsfrage <Frequenz> eine Liste aller Commits des Projekts mit Datum benötigt. Für die zweite Forschungsfrage <Größe> werden detailiertere Informationen über die Commits (LOC / Anzahl Dateien) benötigt. Außerdem ist bei den Commits möglicherweise nicht jede Datei relevant, da es auch Dateien gibt, die nicht zum Build-Prozess gehören (z.B. Anwenderdokumentation).

Bei der Auswahl der Projekte wurde wie folgt vorgegangen:
Es kommen nur Projekte in Frage die

- a) TravisCI benutzen oder benutzt haben, d.h. sie finden sich im TravisTorrent Datensatz wieder
- b) TravisCI nicht zu Beginn des Projekts eingesetzt haben, da sonst kein vorher / nachher Vergleich möglich ist
- c) viel Aktivität verzeichnen können, damit die Datenbasis zur Analyse möglichst groß ist

Es wurde sich für die Projekte rails, openproject und jruby entschieden, da sie die drei größten Projekte gemessen an der Anzahl Builds im TravisCI Datensatz sind. Der genaue Analyseprozess findet sich als jupyter-notebook im Ordner <Overview/Aufgabe4.ipynb / pdf>. Als bereits vorhandene Spalte konnte im TravisTorrent Datensatz das Datum eines jeden Builds ermittelt werden (gh_build_started_at). Über diese Spalte kann Beginn und Ende der TravisCI Nutzung festgestellt werden. Die Frequenz der Commits könnte über GHTorrent, commits, Spalten sha und created_at ermittelt werden, jedoch existieren hier keine detailierteren Informationen über die einzelnen Commits (LOC, Dateien). Daher wurde das Git-Repository analysiert und hier die benötigten Daten zusammengetragen (siehe Teil 2).

Teil 2 - Daten sammeln

Sammlung

Wie in Teil 1 angedeutet, gaben die GHTorrent-Daten die einzelnen commits nicht detailiert genug wieder. Da die ausgewählten Projekte verhältnismäßig groß sind (teilweise über 60000 commits) stellte sich ein Zugriff über die GitHub

API als sehr Zeitaufwändig heraus. Um dennoch die gewünschten Daten zu erhalten, wurden die GitHub-Repositories lokal gespiegelt und anschließend Statistiken wie folgt erzeugt und als CSV aufbereitet.

- Statistik erzeugen: <git log –numstat > outfile>
  Erzeugt eine Textdatei mit Informationen über jedes commit und dateiweise die erzeugten und gelöschten Zeilen.
- Statistik in CSV konvertieren: <java LogfileParser infile>
  Erzeugt CSV-Dateien commits.csv und files.csv aus dem outfile des git Kommandos

Das Java-Tool wurde für diesen Zweck von mir geschrieben und findet sich im Ordner <LogfileParser>

Die Dateien enthalten folgende Spalten:

commits.csv

- <hash> Der eindeutige Hash des commits
- <merge> Optional. Referenz auf die zusammengeführten commits
- <author> Der Autor und i.d.R. dessen Email-Adresse
- <date> Das Datum, Intervallskala (metrisch)
- <desc> Die Beschreibung

files.csv

- <hash> Der eindeutige Hash des commits zu dem die Datei gehört
- <added> Hinzugefügte Zeilen in der Datei, null bei Binary, Verhältnisskala (metrisch)
- <deleted> Gelöschte Zeilen in der Datei, null bei Binary, Verhältnisskala (metrisch)
- <name> Der Dateiname inkl. relativem Pfad
- <ext> Optional. Die Dateiendung, Nominalskala

Die erzeugten csv-Dateien finden sich im Ordner <LogfileParser/data/projektname>.

Vision zur Auswertung

Beiden Forschungsfragen sollen rein deskriptiv mit Boxplots und Histogrammen dargestellt werden. Hier ist fraglich, welche Klasseneinteilung möglichst unbeeinflusste Werte liefert. Daher muss mit verschiedenen Klasseneinteilungen im Bezug auf die Zeit (Commits pro Tag/Woche/Monat) experimentiert werden. Idealerweise zeigt sich mit Umstellung auf TravisCI eine signifikate Veränderung in der Frequenz und / oder der Größe der Commits. Dies kann anschließend noch durch einen Signifikanztest untersucht werden.

Um die Auswertung für mehr als die drei ausgesuchten Projekte durchführen zu können, kann durch ein Skript der Prozess automatisiert werden. Im Folgenden ist die Skript-Idee formuliert, die noch umgesetzt werden muss.

- 1. Projektliste laden <String[]> (Spalte: gh_project_name)
- 2. für jedes Projekt in der Projektliste: <git clone https://github.com/user/project.git> ausführen und das Repository lokal ablegen
- 3. in jedem heruntergeladenen Repository <git log –numstat > outfile> aufrufen um das Logfile zu generieren
- 4. die generierten Logfiles mit LogfileParser in CSV umwandeln
- 5. Optional, die CSV-Dateien mit R auswerten