Dokumentation

Karoline Jüttner

Matrikel-Nr.: 2167763

Github: kajue101

Inhalt

Einleitung

Im Rahmen des Projektseminars I4 "Angewandte Informationswissenschaft" habe ich mir ein Projekt überlegt, das bis zum 23.09.16 fertigzustellen war. Ich habe auf Datahub Statistiken von der App AnkiDroid gefunden, und zwar von den Jahren 2015 und 2016. Dabei geht es u.a. um Informationen zu Installationen pro Land, Installationen pro Sprache und tägliche Abstürze pro Gerät. Auf den genauen Umfang der Statistiken wird im Verlauf der Dokumentation noch genauer eingegangen. Diese Dateien habe ich dann in Python eingelesen, einige Berechnungen angestellt und die gewonnenen Daten mit der Python-Bibliothek Pygal visualisiert. Da ich dabei von meinem ursprünglichen Projektplan etwas abgewichen bin (s. Reflexion des Projektes), habe ich statt der eigentlich geplanten D3 Länderdiagramme als weiteren Teil des Projektes eine Webseite erstellt, auf der die Ergebnisse und diese Dokumentation präsentiert werden sollen. Ich habe mich für dieses Projekt entschieden, da ich im Rahmen des Studiums erst einmal Kontakt mit Datenvisualisierungstools (in dem Fall war es die Bibliothek matplotlib) hatte und gerne weitere Visualisierungsmöglichkeiten kennenlernen wollte. Wie ich bei meinem Projekt vorgegangen bin, wird im Teil "Projektaufbau" genauer erläutert.

Installation

Um dieses Programm zum Laufen zu bringen wird nicht viel benötigt. Folgende Schritte sind eventuell erforderlich:

- Installation von Python 3
- pip install ipython
- pip install pygal
- pip install pygal_maps_world

Wenn die Datei projekt.py ausgeführt wird, werden die Diagramme erstellt. Sie befinden sich dann in dem Ordner "charts".

Was ist AnkiDroid?

Das Wort "Anki" kommt aus dem Japanischen und bedeutet so viel wie "Auswendiglernen" (https://de.wikipedia.org/wiki/Anki). Ein passender Name also, für ein Programm, mit dem man anhand von digitalen Lernkarten Neues erlernen kann. Das Programm ist so konzipiert, dass es sich für verschiedenste Inhalte eignet und das Erlernte im Langzeitgedächtnis gespeichert wird. Diese Inhalte können sowohl selbst erstellt, als auch als fertige Kartendecks heruntergeladen werden (http://www.dennisproksch.de/anki). Damit Anki auch mobil genutzt werden kann, wurde die Andriod-App "AnkiDroid" entwickelt. Es wird empfohlen, die App in Verbindung mit der Desktop-Anwendung zu verwenden, die App funktioniert allerdings auch selbstständig (https://ankidroid.org/docs/manual.html). Der Quellcode der App ist außerdem frei zugänglich auf GitHub zu finden.

Projektaufbau

Die Daten

Die Daten, auf denen mein Projekt basiert, habe ich auf der Organisationsseite von AnkiDroid auf Datahub (Link) gefunden. Die CSV-Dateien beinhalten viele verschiedene Informationen, im folgenden werde ich allerdings nur die nennen, die ich für meine Berechnungen verwendet habe. Die Dateien beinhalten immer entweder Einträge von nur einem Tag oder über einen Zeitraum von einem Monat hinweg. Die Dateien aus dem Jahr 2015 beinhalten Informationen zu momentanen Geräte-Installationen, täglichen Geräte-Installationen, täglichen Geräte-Deinstallationen, momentane Nutzer-Installationen, Nutzer-Installationen insgesamt, tägliche Nutzer-Installationen und tägliche Nutzer-Deinstallationen bezogen auf Länder und Sprachen. Außerdem tägliche Abstürze bezogen auf App-Versionen und verschiedene Geräte, so wie die Durchschnittsbewertung von AnkiDroid in Bezug auf einige Länder. Der Datensatz von 2016 enthält ebenfalls Informationen zu momentanen Geräte-Installationen, täglichen Geräte-Installationen, täglichen Geräte-Deinstallationen, momentane Nutzer-Installationen, Nutzer-Installationen insgesamt, tägliche Nutzer-Installationen und

tägliche Nutzer-Deinstallationen in Verbindung mit Ländern und Sprachen.

Erste Schritte

Erst einmal mussten die verschiedenen CSV-Dateien in Python eingelesen werden. Dafür wurde der Import der CSV-Bibliothek benötigt. Anschließend habe ich für jede Datei eine Liste erstellt, in der sie Reihe für Reihe abgespeichert wird. Da einige der Dateien noch eine Art Dokumentüberschrift enthalten und die Spaltenüberschriften nicht mit in die Listen sollten, habe ich bei einigen dieser Listen die ersten Einträge gelöscht, so dass die erste Teilliste die erste Reihe mit richtigen Daten enthielt. Dann habe ich weitere Listen für jede Spalte, die benötigt wird, erstellt und mithilfe einer Schleife die einzelnen Spalten in die dafür vorgesehenen Listen eingespeichert, durch die Angabe des Index der Spalte. Da alle Einträge in Form von Strings abgespeichert wurden, mussten einige der Listen nun ihre Einträge in Integers umwandeln. Dafür habe ich für jede betroffene Liste eine neue erstellt, in der die Integers gespeichert wurden.

Die Funktionen

Um aus den statistischen Daten Informationen zu gewinnen, mussten ein paar Funktionen geschrieben werden. Zunächst war eine Funktion nötig, die aus einer Liste die höchsten oder niedrigsten Werte heraussucht und so eine Art Rangliste erstellen kann. Die Schwierigkeit dabei war, dass bei einem einfachen Umsortieren der Liste, der Bezug zu anderen Listen verschwindet, da sich der Index ändert. Deshalb habe ich in Zusammenarbeit mit meinem Kommilitonen Philipp Nowak eine Funktion **topX/flopX (liste1,liste2)** (X steht für eine beliebige Zahl) erstellt, die zwei Listen einliest, bei denen der Bezug erhalten bleiben soll und die eine Rangliste von X Einträgen erstellt. Ausgegeben wird eine Liste, die die X höchsten oder niedrigsten Einträge immer abwechselnd mit dem dazugehörigen Element der anderen Liste enthält. In meinem Projekt brauchte ich diese Funktion beispielsweise zur Berechnung der 20 Länder, in denen die App am häufigsten installiert wurde. Die Funktion funktioniert so, dass in einer Schleife das Maximum oder Minimum der ersten Liste ermittelt wird und in die Ergebnisliste kommt. Danach wird der Eintrag mit dem Index des Maximums oder Minimums der ersten Liste in der zweiten ermittelt und ebenfalls an die Liste angehängt. Dann wird entweder der Maximalwert auf 0 (bzw. sehr niedrig) oder der Minimalwert auf

einen sehr hohen Wert gesetzt, damit er im nächsten Schleifendurchlauf nicht mehr berücksichtigt wird. Am Ende wird die Ergebnisliste zurückgegeben und enthält immer abwechselnd ein Element der ersten und zweiten Liste. Das Vorgehen ist auch im Code anhand von Kommentaren recht ausführlich beschrieben. Da die Funktion allerdings die Eingabelisten verändert, müssen diese neu erstellt werden, falls sie noch einmak benötigt werden. Als nächstes war eine Funktion nötig, die das gleiche mit zwei kompletten Listen tut, um sie aufsteigend zu sortieren. Dafür habe ich die Funktion **aufsteigend (liste1,liste2)** geschrieben. Sie funktioniert im Prinzip genau so wie top/flopX, nur dass statt einer Begrenzung der Schleifendurchläufe durch X, eine Begrenzung der Durchläufe mit der Länge der Liste gemacht wird, damit alle Einträge betrachtet werden. Nach jedem Durchlauf werden die verwendeten Einträge aus den Eingabelisten gelöscht, bis diese leer sind und sich geordnet und in abwechselnder Reihenfolge in der Ausgabeliste befinden. Des Weiteren wurde eine Funktion benötigt, die die Summe einer Liste berechnet. Die Funktion **summe (liste)** dazu, geht in einer Schleife alle Einträge durch, addiert sie zur Summe und gibt sie am Ende zurück. Die Funktion **differenz_in_prozent (wert1,wert2)** besteht aus einfacher Prozentrechnung. Die beiden Werte werden zunächst in Float umgerechnet, dann wird der erste Wert durch den zweiten geteilt und das Ergebnis mal 100 gerechnet. Eine weitere einfache Funktion berechnet den Durchschnitt einer Liste. Die Funktion **durchschnitt (liste)** addiert alle Werte, wie bei der Funktion **summe (liste)** und teilt diese Summe dann durch die Länge der Liste. Als letztes gibt es noch die beiden Funktionen **splitlist1(liste)** und **splitlist2(liste)**. Diese wurden benötigt um das Einlesen der Werte in die Diagramme zu vereinfachen. Sie teilen die Listen, die durch die **top/flopX**-Funktion zusammengeführt wurden wieder auf. Die eine Liste nimmt alle geraden Indexe und speichert die Werte in einer neuen Liste und die andere nimmt alle ungeraden.

Berechnungen

Diese Funktionen wurden im Anschluss dann auch für die Berechnungen benutzt. Hier wird nur das Vorgehen beschrieben, die genauen Ergebnisse sind in der Datei projekt.html zu finden. Zunächst habe ich die Funktion **top20** angewandt und zwar mit

den Listen der *Nutzer-Installationen Gesamt* und den *Länderabkürzungen 2015*. Das gleiche habe ich für 2016 gemacht und danach nach dem gleichen Prinzip **flop20** benutzt. Danach habe ich die Top 20 Installationen pro Sprache für 2015 und 2016 aus den Listen *Nutzer-Installationen Gesamt* und *Sprachen* berechnet. Dann die Flop-Werte, wo ich eine Spanne von 40 Einträgen betrachtet habe, aufgrund der vielen Null-Werte. Die Top 20 und Flop 40 Abstürze pro Gerät wurden ebenfalls berechnet, mit den Listen *tägliche Abstürze* und *Geräte*. Anschließend wurde die Funktion **aufsteigend** gebraucht, um die App-Versionen mit ihren dazugehörigen täglichen Abstürzen zu sortieren. Dann wurde die Summe aller Installationen für 2015 und 2016 berechnet, durch addieren aller täglichen Installationen. Hierbei handelt es sich um einen Beobachtungszeitraum von einem Monat. Die Summe der Abstürze im Beobachtungszeitraum von 2015 wurde ebenfalls berechnet. Im Anschluss wollte ich wissen, wie viele Nutzer, die die App installiert haben, sie auch behalten. Dazu wurden alle momentanen Nutzer-Installationen und alle Nutzer-Installationen insgesamt addiert und in die Funktion **differenz_in_prozent** eingelesen. Das wurde sowohl für 2015, als auch für 2016 und beide Jahre gemeinsam berechnet. Daraus konnte man nun auch einfach die Deinstallationsquoten berechnen, indem man den erlangten Wert von 100 abzieht. Als nächstes wurden die täglichen Installationen und die täglichen Deinstallationen jeweils von 2015 und 2016 summiert, um danach die Differenz zwischen den Werten auszurechnen. Außerdem wurde eine weltweite Durchschnittsbewertung berechnet, wofür die Funktion **durchschnitt** benötigt wurde. Zum Schluss kamen dann noch die **splitlist**-Funktionen zum Einsatz, allerdings beziehen sich diese, wie bereits angesprochen, nicht mehr auf die Berechnungen, sondern auf die Diagramme.

Die Diagramme

Für die Umsetzung der Diagramme habe ich mich für die Python-Bibliothek Pygal entschieden, da die Diagramme sehr ansprechend aussehen und z.B. auf Bewegungen mit der Maus reagieren. Man kann zwar nicht mit ihnen interagieren, wie mit einem D3-Diagramm, aber sind dennoch optisch um einiges ansprechender als einfache Diagramme beispielsweise von matplotlib. Benutzt habe ich als ersten ein

Länderdiagramm, das die Länderabkürzungen einliest und die Länder je nach zugehörigem Wert farbig darstellt. Das habe ich einmal für alle Länder gemacht, in Bezug zu den Installationen, einmal für die App-Bewertungen und für die 20 Länder mit den meisten Installationen. Des Weiteren habe ich ein horizontales Balkendiagramm erstellt, das die Installationen 2015 und 2016 im direkten Vergleich darstellen soll. Außerdem wurden einige Balkendiagramme und Kreisdiagramme erstellt. Die verschiedenen Style- und Einstellungsmöglichkeiten habe ich der sehr ausführlichen und verständlichen Pygal-Dokumentation entnommen.

Die Webseite

Zusätzlich habe ich eine Webseite in HTML erstellt und mit CSS angepasst. Dort habe ich Texte zu den Ergebnissen und Diagrammen verfasst und die verschiedenen Diagramme eingebunden um sie präsentieren zu können.

Reflexion des Projektes

Schwierigkeiten

Während der Bearbeitung des Projekts wurde mir klar, dass ich mit D3, was eigentlich zur Darstellung der Länderdiagramme geplant war, nicht rechtzeitig durchkommen würde. Die Erstellung der Diagramme war für mich zu komplex um sie in dem vorgegebenen Zeitraum zu erlernen, zumal ich keine Erfahrungen mit Java-Script besitze und keine hilfreichen Informationen dazu finden konnte, wie man D3 in Verbindung mit Python verwendet. Um dennoch einen gewissen Arbeitsaufwand zu erreichen habe ich beschlossen, die Diagramme mit Pygal zu erstellen, wodurch sie immerhin ein wenig interaktiv geworden sind und zusätzlich die Webseite zu erstellen um die Ergebnisse zu präsentieren.

Was habe ich gelernt?

Aus diesem Projekt werde ich auf jeden Fall den Umgang mit Pygal mitnehmen, was mir viel Spaß gemacht hat, da Pygal sehr viele Gestaltungsmöglichkeiten bietet. Mit Pygal würde ich auch gerne in Zukunft noch einmal arbeiten und ich bin froh, dass ich auf diese Binliothek gestoßen bin. Des Weiteren konnte ich nach dem Seminar "Strukturieren digitaler Dokumente" im ersten Semester meine HTML- und CSS-Kenntnisse noch einmal auffrischen. Auch den Umgang mit der Versionierung mit Github habe ich nun noch einmal verinnerlicht.

Fazit

Insgesamt werde ich einiges aus der Bearbeitung dieses Projektes mitnehmen, auch wenn ich nicht alles geschafft habe, was ich mir vorgenommen habe und ein wenig von meinem Projektplan abgewichen bin. Dennoch bin ich mit dem Projekt und den entstandenen Diagrammen so wie der Webseite im Großen und Ganzen zufrieden.

Literaturverzeichnis: