README für Eye-Tracking Dashboard (app.py)

Einführung

In diesem Projekt wurde ein Dashboard mithilfe der Python-Bibliotheken Plotly und Dash entwickelt, das einen Eyetracking-Datensatz aus einer Studie zu Metrokarten verschiedener Städte analysiert. Der Schwerpunkt der Analyse liegt auf dem Vergleich zwischen farbigen und graustufigen Karten. Das Dashboard bietet verschiedene Visualisierungstools, um die Stimuli aus unterschiedlichen Blickwinkeln zu betrachten und zu analysieren.

Zielgruppen und Datenquelle

Das Dashboard richtet sich primär an Forscherinnen und Forscher aus dem Bereich Eye Tracking. Es bietet eine Plattform zum Vergleich der Effektivität farbiger und graustufenbasierter Kartendarstellungen. Ziel ist es, Studien bei der Entscheidungsfindung zu unterstützen, ob graustufen Karten genauso effektiv sind wie farbige Karten.

Der verwendete Datensatz *'all_fixation_data_cleaned_up'* (im Ordner 'assets') wurde von Dr. rer. nat Michael Burch (Fachhochschule Graubünden) zur Verfügung gestellt und stammt aus seinen Forschungsarbeiten im Bereich Eye-Tracking.

Datenaufbereitung

Der Datensatz wird als CSV mit ';' als Trennzeichen mit Pandas eingelesen. Das Dataframe hat die folgende Struktur:

- **Timestamp:** Identifikation für den Zeitpunkt des gemessenen Datenpunktes (ist für das Programm nicht weiter relevant).
- **StimuliName:** Bezeichnung der Metro-Karte, z.B. '01_Antwerpen_S1.jpg'. Jede Metrokarte kommt in vier Ausprägungen vor: 2x in graustufen und 2x in farbe). Zu jeder Karte gibt es eine Version S1 und S2, wobei die zweite Version um 90° bzw. 180° gedreht ist. Die graustufen Karten sind mit 'b' kekennzeichnet.
- **FixationIndex:** Eine fortlaufende Nummeriereung, welche die Reihenfolge angbigt, in der die Blickpunkte aufzeichnet wurden.
- **FixationDuration:** Die Dauer des Blickpunkts an einer bestimmten X-Y-Koordinate in ms.
- **MappedFixationPointX:** Die X-Koordinate des Blickpunktes auf der Metrokarte.
- **MappedFixationPointY:** Die Y-Koordinate des Blickpunktes auf der Metrokarte.
- **User:** Probanden der Studie. Es gibt 40 Studienteilnehmer im Datensatz, P1-P40.
- **description:** Enthält die dichotome Variable 'grey' oder 'color'.
- **CityMap:** Gleich wie StimuliName nur ohne Prefix vor dem Städtenamen, um nicht zwischen grau und farbig zu unterscheiden, z.B. 'Antwerpen_S1'. Diese Variable wird für das Dropdown Menü verwendet und enthält jeweils eine Ausprägung S1 oder S2.

- **City**: Aggregation aller Ausprägungen einer Stadt (S1 und S2 und grau und farbig), z.B. ‘Antwerpen’. Diese Variable wird in der globalen Übersicht verwenden (Boxplot).
- **SaccadeLength**: Euklidischer Abstand zwischen zwei Fixationspunkten.

Anhand dieser Initialdaten werden im Programm app.py weitere Variablen berechnet:

- **Taskduration** = FixationDuration aggregated = Summe aller Fixationspunkte eines Probanden und Stimulus.
- **Average Taskduration** = Mittlere Taskduration aller Probanden für einen Stimulus
- **Average FixationDuration** = Mittlere Fixationsdauer aller probanden für einen Stimulus

Datenverwendung

Das Dashboard ermöglicht eine Analyse anhand verschiedener Visialisierungen in zwei Dimensionen:

- **Globale Analyse:**
  - KPI-Tabelle
  - Histogramm
  - Boxplot
  - Korrelation bzw. Scatterplot
- **Detailanalyse:**
  - KPI-Tabelle
  - Histogramm
  - Gazeplot zur Scanpfad-Analyse
  - Heatmap zur Density-analyse
  - Korrelation bzw. Scatterplot

Die Auswahl des Visualisierungstyp erfolgt über einen Click-Button. In der Detailanalyse muss zudem die gewünschte CityMap über das Dropdown Menü gewählt werden.

Technische Voraussetzungen

Um den Code *‘app.py’* auszuführen, ist die Installation von Python Version 3.10 oder höher erforderlich. Zudem müssen folgende Packages installier und importiert werden:

```
from dash import Dash, dash_table, dcc, html, Input, Output, State, callback_context
from dash_iconify import DashIconify
from PIL import Image
import dash_bootstrap_components as dbc
import pandas as pd
import plotly.express as px
import glob
import plotly.graph_objects as go
```

Installation und Verwendung der Applikation

1. Repository klonen (Link zum Git Repository)

```
git clone <https://github.com/TamaraFHGR/Consultancy-Project-1.git>
```

2. Obige Packages installieren (sofern nicht bereits vorhanden):

```
pip install dash dash_iconify dash-bootstrap-components plotly pandas
```

3. Es muss sichergestellt sein, dass das Verzeichnis '*assets*' angelegt ist und die Datasource '*all_fixation_data_cleaned_up.csv*' und das Stylesheet '*custom.css*' im Verzeichnis vorhanden sind. Zudem müssen alle jpg.-Dateien der 24 Städte im Ordner enthalten sein (24 x 4 = 96 jpg-Files).

4. Starten der Anwendung '*app.py*' mit Python:

```
python app.py
```

5. Die Dash-Anwendung öffnet sich automatisch im lokalen Webbrowser mit dem URL http://127.0.0.1:8050/.

Öffentlicher Website Zugang

Das Dashboard ist auf der folgenden Website deployed und kann über den Link aufgerufen werden: https://tamarafhgr.pythonanywhere.com/

Codeaufbau

In diesem Projekt wird Python als Open-Source-Framework zur Erstellung einer reaktiven Webanwendungen verwendet. Dies ermöglicht es, Python-Code für die funktionalen Komponenten zu schreiben und die Designkomponente in ein CSS (Cascading Style Sheets) auszulagern.

- **Dash Core Components**: dcc-Komponenten bilden das Grundgerüst des Dashboards und ermöglichen interaktive Elemente, wie z.B. Dropdown-Menüs (Wahl zwischen Light und Dark Mode, Auswahl der City Map sowie Auswahl der User), Range-Slider (Einschränkung der Taskduration) oder die Aktivierung der Click-Buttons.
- **Dash HTML Components**: HTML-Komponenten werden verwendet, um die Struktur und das Layout der Dash-Anwendung zu definieren. Verschiedene html.Div Elemente dienen als Container um Inhalte zu gruppieren. Aber auch für Header und Textelemente werden HTML-Komponenten eingesetzt.
- **CSS (Cascading Style Sheets)**: Das File *'custom.css'* im Ordner *'assets'* wird genutzt, um das visuelle Erscheinungsbild des Dashboards zu gestalten. Grundsätzlich ist das Layout in drei Spalten organisiert.
- **Plotly-Diagramme:** Die Bibliothek Plotly wird genutzt, um interaktive Diagramme darzustellen, die den Datensatz im Dashboard visualisieren. Jedes Diagramm ist in einem html-Container organisiert und ist durch eine Funktion definiert ('def update_graph_X').
- **Callbacks:** Callbacks werden verwendet, um die Dash Core Components und Plotly-Diagramme miteinander zu verbinden und eine Interaktion mit dem Benutzer zu ermöglichen. Jeder '*def-Funktion*' geht ein Callback voraus, der den Output und Input der Funktion bestimmt.

Autoren

Das Dashboard wurde im Auftrag der Fachhochschule Graubünden (FHGR) im Masterstudiengang für Data Visualization entwickelt und am 20.06.2024 zum letzten Mal aktualisiert. Die Autoren sind:

- Nyffeler Tamara
- Pellegatta Serge
- Reiser Sharon