Assignment 4 Report

EE23B039 A Lekhasree

Approach

Keyboard Layout Selection

First input is taken to select the desired keyboard layout. Depending on the layout chosen, the keys in all rows are initialized as empty strings. Subsequently, the variables `row1_keys`, `row2_keys`, `row3_keys`, and `row4_keys` are populated with the characters present on the keyboard, other than special characters such as `Enter`, `Shift`, and others.

Handling Special Characters

Following this, a list of special characters is made to capture the names of these keys, including `Tab`, `CapsLock`, `Enter`, and `Shift`. A `keyboard_layout` is then constructed, using a standard keyboard as a reference. The widths of the keys are determined through intuition and trial and error, so that the keys are not overlapping and they are positioned correctly.

Layout Compatibility

It is observed that for different keyboard layouts, such as `QWERTY`, `Dvorak`, and `Colemak`, the primary distinction lies in the arrangement of characters. The layout and special characters remain constant. So because of this i fixed the layout and special characters, only letters and numbers had to be changed for each keyboard layout.

Shift Key Activation

To decide which `Shift` key should be pressed, a dictionary is established that maps each key on the keyboard to its respective shift character. A function is implemented to determine the appropriate `Shift` key: if the x-coordinate of the key is less than half the total keyboard width, the right `Shift` will be engaged; otherwise, the left `Shift` will be activated. This design choice is made based on the intuitive assumption that it minimizes finger travel distance.

Number of times a certain character is pressed

A dictionary is created to track the number of times each key on the keyboard is pressed. Utilizing the values recorded in this dictionary, a heat map is generated to visualize key usage. if a capital letter or a shift character like !, @ or A G J are pressed, then one of the shifts based on the above function and the letter to be pressed along with shift to get the shift character both are incremented by 1 in this dictionary.

Heat Map Generation

The heat map generation process involves plotting rectangles that correspond to the layout of the keyboard. Each rectangle's color is determined by normalizing the count of key presses (by dividing with the maximum value), with the resulting color filled into the respective rectangle.

Finger Travel Distance Calculation

To compute the total finger travel distance, specific home row indices are designated (0, 1, 2, 3, 6, 7, 8, 9). The home keys comprise the letters located in row 3 corresponding to these indices (in qwerty layout home keys are a, s, d, f, j, k, l and ;) . For any character input, the nearest home key is identified by checking for the first home key within a specified range of 1.5 units along the x-axis.

If the key belongs to the home row, the finger travel distance is recorded as zero. Otherwise, the distance between the character and the nearest home key is calculated and accumulated in the total distance, which

was initialized to 0. **This distance is multiplied by two, based on the assumption that the finger returns to the home key after typing each character.**

Sample tests:

Normalization of frequencies is done by dividing all the frequencies with maximum frequency.

Test case 1:

Keyboard Layout

The qwerty keyboard layout is used for this test case.

Text Entered

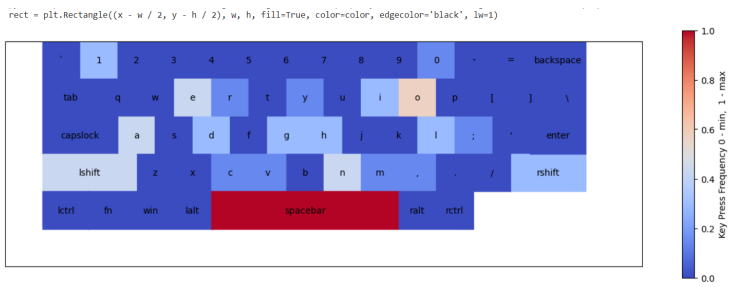
```
Enter a string:  Hello!!  good morning, have a nice day:)
```

Frequency Dictionary

The following frequency dictionary shows the number of times each key was pressed:

```
{'lshift': 3, 'shift': 5, 'h': 2, 'e': 3, 'l': 2, 'o': 4, 'rshift': 2,
'1': 2, 'spacebar': 7, 'g': 2, 'd': 2, 'm': 1, 'r': 1, 'n': 3, 'i': 2,
',': 1, 'a': 3, 'v': 1, 'c': 1, 'y': 1, ';': 1, '0': 1}
```

Heat Map Visualization



Finger Travel Distance

The total finger travel distance, calculated based on the input string, is: 72.16766515910885 units

Test Case 2

Keyboard Layout

The Dvorak keyboard layout is used for this test case.

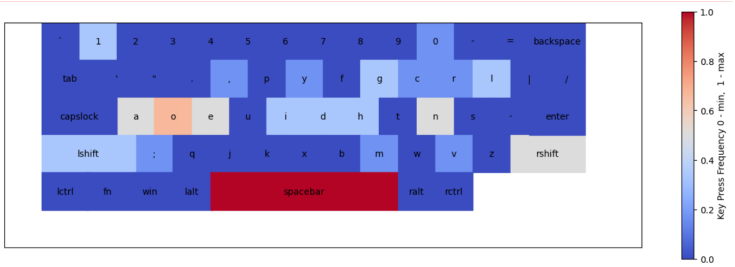
Text Entered

```
Enter a string:  Hello!!  good morning, have a nice day:)
```

Frequency Dictionary

The following frequency dictionary shows the number of times each key was pressed:

```
{'lshift': 3, 'shift': 5, 'h': 2, 'e': 3, 'l': 2, 'o': 4, 'rshift': 2,
'1': 2, 'spacebar': 7, 'g': 2, 'd': 2, 'm': 1, 'r': 1, 'n': 3, 'i': 2,
',': 1, 'a': 3, 'v': 1, 'c': 1, 'y': 1, ';': 1, '0': 1}
```



Heat Map Visualization

Finger Travel Distance

The total finger travel distance, calculated based on the input string, is: 53.313627197893894 units

Test Case 3

Keyboard Layout

The colemak keyboard layout is used for this test case.

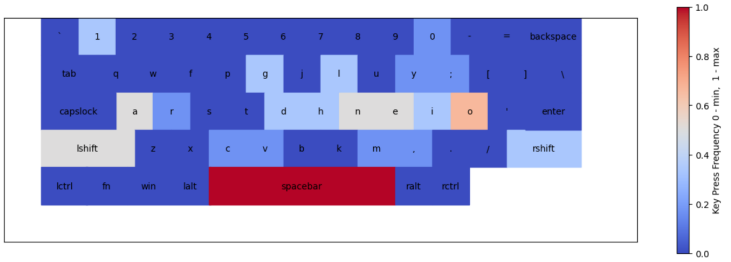
Text Entered

```
Enter a string:  Hello!!  good morning, have a nice day:)
```

Frequency Dictionary

```
{'lshift': 3, 'shift': 5, 'h': 2, 'e': 3, 'l': 2, 'o': 4, 'rshift': 2,
'1': 2, 'spacebar': 7, 'g': 2, 'd': 2, 'm': 1, 'r': 1, 'n': 3, 'i': 2,
',': 1, 'a': 3, 'v': 1, 'c': 1, 'y': 1, ';': 1, '0': 1}
```

Heat Map Visualization



Finger Travel Distance

The total finger travel distance, calculated based on the input string, is: 52.90963804114631 units

From the above values of finger travel distance, for the string Hello!! good morning, have a nice day:), colemak is the most efficient one as it has the least finger travel distance.

BONUS:

The initial part of the code is just like the above one, asking to choose which keyboard layout and designing the layout. Then, when a text is entered, it breaks it down letter by letter. Each letter has a "position" on the keyboard. The script uses matplotlib to draw the keyboard with rectangles for each key and updates it for each key press from your input. The animate function is written which checks if a key needs Shift (like capital letters or symbols) and highlights both the Shift key and the key from the string. If no Shift is needed, it just highlights the pressed key. When a key is pressed it only colours that particular rectangle (according to width and position from layout) and others are left white. So this way an animation is created using FuncAnimation and displayed.