APL Assignment 4

Arivoli Ramamoorthy

October 1, 2024

Contents

1 Introduction

This report covers my implementation of generating a keyboard heatmap visualization of the key usage and calculating the total distance traveled by fingers while typing to analyze the efficiency of keyboard layouts.

2 Detailed Function Analysis

2.1 calculate_distance

```
def calculate_distance(char, layout):
    keys = layout["keys"]
    characters = layout["characters"]

```

```
    def distance(start, end):
        start_pos = keys[start]["pos"]
        end_pos = keys[end]["pos"]
        return math.sqrt(
            (start_pos[0] - end_pos[0]) ** 2 + (start_pos[1] -
    end_pos[1]) ** 2
        )

    key_sequence = characters.get(char, [])
    total_distance = 0

    for key in key_sequence:
        start_key = keys[key]["start"]
        total_distance += distance(start_key, key)

    return total_distance
```

This function calculates the distance a finger travels to type a single character:

- It uses the layout's key and character mappings.
- For each key in the character's sequence, it computes the == distance from the starting position to the key position.
- It sums these distances to get the total travel for the character.
- It considers uppercase letters and special characters (use of Shift key), and adds that distance to the total distance as well.

2.2 analyze_text

```
def analyze_text(text, layout):
    key_usage = {key: 0 for key in layout["keys"]}
    total_distance = 0
    key_sequence = []

    for char in text:
        if char in layout["characters"]:
            for key in layout["characters"][char]:
                key_usage[key] += 1
                key_sequence.append(key)
            total_distance += calculate_distance(char, layout)

    return key_usage, total_distance, key_sequence
```

This function processes the input text:

- It initializes counters for key usage and total distance.
- For each character in the text:
  - It updates the key usage count.
  - It calculates the travel distance using `calculate_distance`.

  - It records the sequence of keys pressed.
- It returns the key usage, total distance, and key sequence.

2.3 generate_heatmap

```
def generate_heatmap(layout, key_usage):
```

This function creates a visual heatmap of key usage:

- It sets up a matplotlib figure.
- It creates a custom color map from white to red.
- It uses a grid and Gaussian blur for smooth color transitions.
- For each key in the layout:
  - It draws the key as a rectangle.
  - It adds the key label.
  - It overlays a colored circle representing usage intensity.
- It applies Gaussian blur to create a smooth heatmap effect.
- It sets the plot limits, title, and aesthetics.

2.4 animate_typing

```
def animate_typing(layout, key_sequence):
```

This function generates an animation of the typing process:

- It sets up a matplotlib figure.
- It defines initialization and update functions for the animation.
- It uses matplotlib's `FuncAnimation` to create the animation.
- It highlights each key press in sequence with a red circle.
- It returns the animation object and figure for saving.

3 Outputs

The code uses two main visualization techniques and also outputs total finger travel distance:

3.1 Distance travelled by the finger

- This is calculated by calling the function `calculate_distance` under the function `analyze_text`.
- It includes the travel distance of Shift-key for uppercase letters and some symbols.

3.2 Heatmaps

- The heatmap is created using matplotlib's `Rectangle` and `Circle` patches
- Color intensity represents key usage frequency
- Uses Gaussian blur for smooth color transitions
- Provides a static overview of key usage patterns

3.3 Animations

- The animation is created using matplotlib's `FuncAnimation`
- Shows the sequence of key presses over time
- Highlights each key press with a red circle
- Provides a dynamic view of typing patterns

4 Layout Info

The keyboard layouts are specified as Python dictionaries with two main components:

1. `keys`: A dictionary mapping each key to its position and starting position.
2. `characters`: A dictionary mapping each character to the sequence of keys needed to type it.

I have used the layout that sir gave for the programming quiz, and I used chatgpt to generate the layout files for Dvorak and Colemak.

5 Usage Instructions

To use the keyboard layout analyzer:

1. Run the program by using the command

```
python3 ee23b008.py
```

2. When prompted after running the program, enter the text that is to be analyzed.

3. The script will generate heatmaps and animations(gifs) for each layout (QWERTY, Dvorak, Colemak).

4. Review the output to compare the efficiency of different layouts for your input text.

6 References

- Use of chat-gpt for generating the Dvorak and Colemak layout files and for looking up library functions.
- Discussion with classmates about heatmap generation, gaussian blur for smoothness.
- Google, python docs for libraries and their functions.