Assignment 4: Keyboard Analysis

October 2, 2024

1 Introduction

The goal of this project is to generate a heatmap overlay on a keyboard layout based on a provided text input, visualizing the frequency of key usage. Additionally, we calculate the total distance traveled by the fingers while typing the input text. This analysis can be useful for ergonomic design and typing efficiency evaluation.

2 Approach

The approach I adopted involves:

1) Keyboard Layout as a Grid:

- The keyboard layout is defined as a grid where each key is assigned coordinates that approximate their positions on a standard QWERTY keyboard. Rather than using the actual physical dimensions of a keyboard, I mapped the layout to a simplified grid structure, with each key given an (x, y) coordinate. This makes it easier to generate the layout programmatically. Special keys like the spacebar and shift keys are given custom dimensions to reflect their physical size.

2) Generating the Keyboard Layout Image:

- The keyboard layout is rendered using matplotlib by plotting each key at its respective (x, y) position. The layout is then saved as an image using canvas.draw(), which renders the entire figure into an image buffer. This figure is converted into a NumPy array using canvas.tostring_rgb() and reshaped to a standard format. This allows the keyboard layout to be used as a background image for further overlays (such as the heatmap), without having to store an intermediate image file.

3) Heatmap Overlay on the Layout:

- Key usage frequency data is collected and visualized using a heatmap that overlays the keyboard layout. The plt.imshow function is employed to plot the heatmap. For this, I use an RGBA color scheme, where transparent values (alpha=0) are assigned to keys that are not used in the input text. This allows unused keys to remain transparent, showing the keyboard layout underneath, while the used keys are colored based on their usage frequency.

- The heatmap interpolation uses spline36, which applies cubic spline interpolation to create smooth transitions between the frequency data points. This results in a more visually appealing and continuous gradient.

4) Finger Travel Distance Calculation:

- In addition to visualizing key usage, I also calculate the finger travel distance for the given input text, which is printed with an accuracy of upto 3 decimals. For each key press, I calculate the Euclidean distance between the home key of the finger and the target key. For characters involving shift, I make sure to use the shift on the other side of the keyboard as the character, to get the least distance.

3 Section Breakdown

3.1 Section I. Generating the keyboard layout as an image

In this section, the keyboard layout is rendered and saved as an image using matplotlib. The layout is plotted with some special keys such as Space, Shift_L, and Shift_R given custom dimensions to reflect their actual physical size on a keyboard. After rendering, the figure is saved as a NumPy array via a canvas, allowing it to be used directly in other parts of the analysis without saving it as an image file on disk.

3.2 Section IV. Generating heatmap as an overlay on the layout

In this section, the heatmap is generated based on the frequency of key usage from the input text. matplotlib is used to overlay a semi-transparent heatmap on the keyboard layout. The color gradient indicates key usage intensity, with blue for rarely used keys and red for frequently used keys. The spacebar is intentionally excluded from the heatmap, but can be included by modifying the code.

The plt.imshow function is employed with interpolation='spline36', which refers to a cubic spline interpolation. This method ensures smooth transitions between key usage values, providing a more visually coherent heatmap compared to rigid circles.

3.3 Section V. Extracting the home row from the given layout

To calculate finger travel distance, I extracted the home row keys for both hands. The home row for QWERTY is a, s, d, f for the left hand and j, k, l, ; for the right hand. This extraction helps identify which hand is responsible for typing each key, and the function is flexible to work with layouts like Colemak and Dvorak. I also save the location (right or left side of the keyboard) of the home key being used for typing a letter with Shift involved.

3.4 Section VI. Calculating total distance travelled by the fingers

This section computes the total distance traveled by fingers during typing. For each character in the input text, it calculates the distance between the finger's home key and the target key, summing these distances to give a total for the entire text.

For uppercase letters or symbols, I use the opposite side's Shift key (e.g., left Shift for right-hand letters), as mentioned in the document regarding calculating distance when Shift is involved.

4 Findings

- Heatmap Visualization of Key Usage: The heatmap provides clear insight into key usage patterns for the input text. Frequently used keys are easily identifiable due to their red

coloring, while rarely used keys appear blue, and unused keys remain transparent. This visualization helps highlight which parts of the keyboard are used most and whether certain areas are underutilized. For instance, letters like 'e', 'a', and 's' are typically used more frequently, making their corresponding keys turn red in the heatmap.

- Finger Travel Efficiency: By calculating the total finger travel distance for the input text, it is possible to assess the ergonomic efficiency of typing on different layouts. For example, in the QWERTY layout, fingers may need to travel further distances due to the positioning of frequently used keys. The function efficiently tracks the travel from the home row keys and highlights which hand and finger are used, providing valuable data on how a layout impacts finger movement. This analysis can be extended to evaluate alternative layouts like COLEMAK or Dvorak.
- Grid-Based Layout vs. Physical Keyboard Layout: The grid-based representation of the keyboard simplifies the rendering process but does not fully capture the physical layout and relative distances of a real keyboard. While this abstraction works well for visualizing usage frequency, it may introduce slight inaccuracies in the finger travel distance calculation compared to a more physically accurate model. However, the general trends in usage and movement remain consistent and useful for analysis.
- Impact of Spline Interpolation: The use of spline36 interpolation for the heatmap smooths out the transitions between key usage frequencies, creating a more visually appealing representation. This choice is beneficial for identifying overall typing patterns rather than focusing solely on individual key presses.

5 Bonus Challenge: Implementing Colemak, Dvorak and comparing efficiency

My findings for the challenge were the following:

- After implementing and analyzing heatmaps for QWERTY, Dvorak, and COLEMAK layouts using two different input texts, I observed that COLEMAK $<$ Dvorak $<$ QWERTY in terms of finger travel distance, with the difference in distance between QWERTY and COLEMAK sometimes reaching 200 units.
- The QWERTY layout exhibited a much wider spread in the heatmap compared to Dvorak and COLEMAK, indicating that frequently used keys are distributed across the keyboard. This contrasts with the more compact heatmap in Dvorak and COLEMAK, where key presses are concentrated in specific areas.
- Both Dvorak and COLEMAK showed most of the high-frequency keys on the home row, contributing to their reduced finger travel distance. This home row optimization explains their superior typing efficiency compared to QWERTY.
- Pictures have been included in the .zip file, which show the heatmaps and travelling distances for each keyboard layout for 2 different input texts.