KeyNet: keyboard layout optimization using network analysis and genetic algorithms

Anže Arhar, Kristjan Kostanjšek, and Nejc Ločičnik

1 Introduction

As we know, typing speed is one of the most important quality measures for every dedicated computer science engineer. A significant factor influencing typing speed is the keyboard layout used. The most popular keyboard layouts today include QWERTY, AZERTY, Dvorak, and Colemak. Of these, QWERTY remains the dominant layout, despite its origins dating back to the 1870s. While QWERTY has been incrementally improved over time, resulting in a relatively optimal layout given its historical constraints, there is potential for entirely different layouts that could significantly enhance typing speed and comfort. Moreover, the optimal keyboard layout can vary depending on the specific typing task, such as coding versus writing a novel. The layout can also vary significantly depending on the language being typed.

To address the challenge of optimizing keyboard layouts, we employed an ortholinear keyboard. We began by utilizing network analysis to construct a weighted directed graph, where the nodes represented keys and the links represented the frequency of successive key presses. With the analysis of this graph we built the foundation for our initial layout. Subsequently, we applied genetic algorithms to refine and optimize this layout to the greatest extent possible. Although we did not have sufficient time to personally become fully proficient with the new layouts, we evaluated them using other metrics, such as the distance our fingers traveled while typing various texts.

This paper aims to optimize keyboard layouts tailored to specific types of text, including programming code, English language text, Slovenian language text, and more. Additionally, we seek to evaluate the efficiency of the layouts we developed and compare them to current popular keyboard layouts, such as QWERTY and Dvorak. By doing so, we hope to identify layouts that offer superior performance for specific tasks, ultimately enhancing typing speed and comfort for users.

2 Methods

In this section, we will detail the process of designing the keyboard layout. First, we gather text samples (e.g. Wikipedia articles and programming code) and preprocess them by removing unnecessary symbols, spaces, line breaks, and tabs. This preprocessing results in a string composed solely of the 26 English alphabet letters and 4 additional symbols: period, comma, hyphen, and a colon. This gives us a total of 30 characters, suitable for our 3 x 10 ortholinear keyboard. Next, we use this preprocessed text to construct a graph, employing network analysis methods to establish the foundation of our keyboard layout. Finally, this foundational layout is subjected to a genetic algorithm to achieve full optimization.

2.1 Network analysis

TODO

2.2 Genetic algorithms

As key layout optimization is a permutation problem, we decided to optimize our layout using a genetic algorithm. By defining the problem in this way, we can easily design a cost function that is defined only using matrix operations. This allows us to speed up the computations significantly, resulting in better layout designs.

Each gene in the population is represented as a permutation. Our goal is to find the optimal permutation of keys that minimizes the cost function c. In this section, the function diag refers to creating a diagonal matrix from a vector $v \in \mathbb{R}^n$:

$$\text{diag}(v) = D \in \mathbb{R}^{n \times n}, \text{where } D_{i,j} = \begin{cases} v_i, i = j \\ 0, \text{ otherwise} \end{cases}$$

We define the cost function c using the following matrices:

- Probability matrix P contains the bigram probabilities

$$P_{i,j} = p_{i,j}$$

- Markov chain transposition matrix A of the network is used to calculate the stationary probability vector π

$$\pi A = \pi$$

- Preferred position matrix R is a diagonal matrix of key importances

$$R = \text{diag}\left(\text{vec}\left(\begin{bmatrix} 2 & 3 & 4 & 5 & 1 & 1 & 5 & 4 & 3 & 2 \\ 6 & 7 & 8 & 9 & 2 & 2 & 9 & 8 & 7 & 6 \\ 2 & 3 & 4 & 5 & 1 & 1 & 5 & 4 & 3 & 2 \end{bmatrix}^T\right)\right)$$

- Distance matrix D contains the physical distances between the keys

$$D_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

- Same finger bigram matrix F groups keys assigned to the same finger

$$g = \text{vec}\left(\begin{bmatrix}1&2&3&4&4&5&5&6&7&8\\1&2&3&4&4&5&5&6&7&8\\1&2&3&4&4&5&5&6&7&8\end{bmatrix}^T\right)$$

$$F_{i,j} = \begin{cases}1, g_i = g_j\\0, \text{otherwise}\end{cases}$$

Cost c is defined using a weighted sum of the permuted matrices which is finally element-wise summed into a single number

$$C(E) = EP \odot (w_1 F + w_2 D) - w_3 \text{diag}(E\pi) R$$

$$c = \sum_i \sum_j C_{i,j}$$

Using a weighted sum allows us to assign more weight to some scores and less to other. All of the aforementioned matrices are displayed in Figure 1.

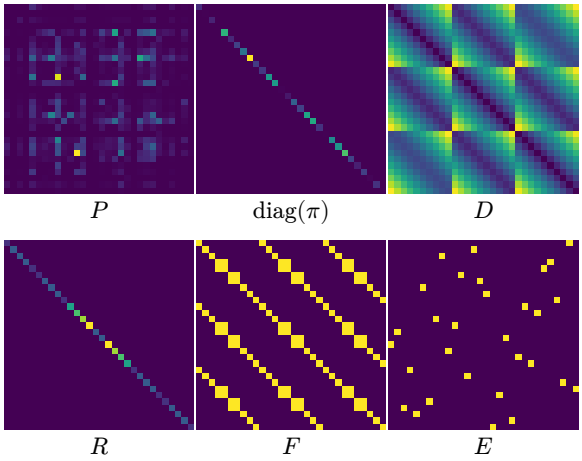


Figure 1: Matrices visualized

In each generation we keep the 10 % of the previous population with the lowest cost. Next 50 % of the genes according to their cost are only mutated. Mutations are represented using a random swap of two elements. Other 40 % of genes are a recombination of all genes from the previous population. This is achieved using the partially mapped crossover (PMX) algorithm, which swaps parts of two permutations in a way that keeps a part of the original permutation. Candidates for recombination are selected according to their cost - lower cost has a higher probability of being selected and higher cost has a lower probability of being selected. The mutation and crossover procedure operate only inclusively on home-row keys or other keys. This limtation is added to keep similar structure to the layout designed using network analysis.

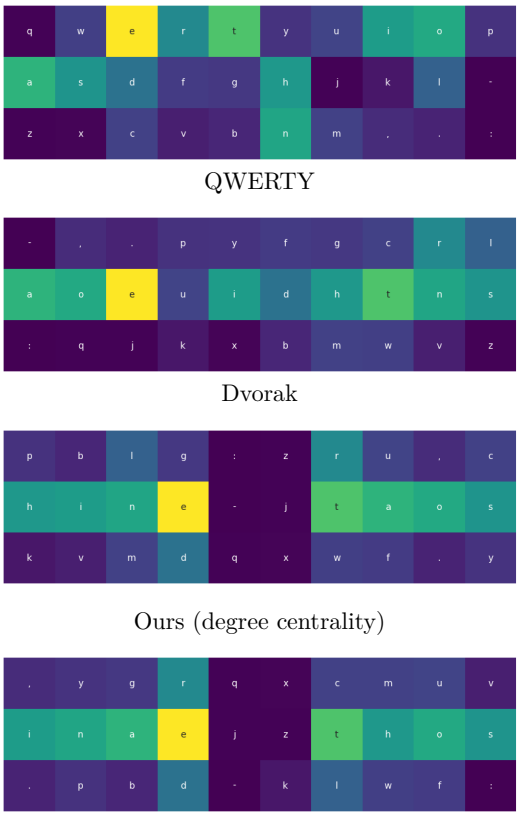
2.3 Evaluation

We constructed our keyboard layouts using text from War and Peace [1] by Leo Tolstoy. Evaluating the effectiveness of these keyboards presents a unique challenge. One potential method of evaluation involves using the cost function described in Section 2.2. However, this approach could introduce bias, as the keyboards were optimized based on this very cost function.

To avoid this bias, we decided to take a different approach. We visualized each keyboard layout with a heatmap representing the frequency of key presses. By analyzing these heatmaps, we can gain insights into the distribution of key usage and provide qualitative commentary on the efficiency and comfort of the layouts.

3 Results

TODO



QWERTY

Dvorak

Ours (degree centrality)

Ours (genetic algorithms)

Figure 2: Keyboard layouts visualized using a heatmap

4 Discussion

Because of time limitations we managed to test it only on the War and Peace novel, but we've developed a framework where we could easily test it on different types of text as well (e.g. programming code).

Bibliography

[1] Tolstoy Leo, *War and Peace*, vol. 6. 1869.