What makes a keyboard layout "efficient" ?

Gia Luong Vu

January 22, 2025

Abstract

Typing has been one of the most popular ways to write down any piece of information, and a keyboard layout is probably the most important factor. However, QWERTY has been proven many times to be less inefficient than others such as DVORAK or COLEMAK. This research aims to find the factors that are taken into account to make a layout efficient. Through an evolutionary algorithm called a genetic algorithm, I have found some characteristics ....

1 Introduction

QWERTY was developed during the late 1870s from the piano-like alphabetic keyboard, as an adaption to the morse code and partly the jamming problems. Thus, many large manufacturers adopted and produced keyboards with this layout.

This layout has been used since then, although many hardware issues have been resolved for years. Thus, it is intuitive that this layout might not be suitable for the current situation. However, as it does not affect much in people's lives, not mentioning learning a whole new layout takes a lot of time, QWERTY is still now the main layout used in daily life.

Later, DVORAK and COLEMAK layout was developed to reduce finger movement. DVORAK, or Dvorak Simplified Keyboard, was created around the 1930s by Dvorak and his brother while COLEMAK was introduced later, in 2006, by Shai Coleman. Its purpose is to minimize learning curve, while still achieve certain effectiveness from DVORAK.

While this paper only solve the keyboard problem for English language only, the application can actually solve the same problem for other languages given that the alphabet is interchangeable.

1.1 Objectives

The primary objective of this research is to identify and analyze the characteristics that contribute to the efficiency of a keyboard layout using Genetic Algorithms. To achieve this goal, the study will focus on the following specific objectives:

1. Develop and Implement a Genetic Algorithm: Design and implement a Genetic Algorithm tailored for optimizing keyboard layouts.

2. Evaluate Layout Efficiency: Assess different keyboard layouts based on key metrics such as typing speed or accuracy.

3. Identify Key Characteristics of Efficient Layouts: Analyze the results to determine the key features and characteristics that contribute to an efficient keyboard layout.

4. Compare with Existing Layouts: Given the characteristics, generate a keyboard from that rule and compare to other layouts like QWERTY or DVORAK, making sure that the factors does contribute to the effectiveness of the layout.

By fulfilling these objectives, the research aims to provide valuable insights into the design of efficient keyboard layouts.

2 Methodology

2.1 Research Design

The idea is to create 2 main applications: the first one is the genetic algorithm, which produce the most efficient keyboard layout; the second one is a GUI (Graphical User Interface) that run the simulation of typing visually. From that, spot the characteristic of those keyboard layouts, categorize them in features.

The aim of the model is to find about 3 to 4 categories that a keyboard layout can fall into in order to measure it.

2.2 Genetic Algorithm

Genetic algorithm is an evolutionary algorithm, which means that it does not generate a solution, but utilize mechanisms from the biological evolution, of human, or ants for example. In this case, genetic algorithm bases on the natural selection of any population, consisting of phases such as selection, crossover (reproduction) and mutation.

The next parts will specified how my EA adapted to the keyboard layout's efficiency problem.

Initial Population: There will be a random keyboard generator that create a randomized keyboard. The starting population will be about 100 - 500

2.2.1 Fitness function

Evaluating criteria: I have done some research on this part, and a large number of people use total distance of their metrics. However, I believe that the total distance will not enable different finger speed as it is apparent that index finger will be faster than the pinky finger. This can create a large difference in keyboard design, for instance, for key in the home rows, frequently used key should be placed in the middle, not outside. And it is also untrustworthy to use time as the metric since texts have different length.

My metric: Therefore, I'm choosing letter per second (LDS) as my metrics, as it is more reliable than word per second (Some words have lots of letter while some has little). Therefore, the algorithm is trying to maximize this fitness score

In this research, I'm ignoring the difficulty of distinct hand positions, which make typing much harder.

Implementation: LDS is calculated using a typing test simulation that can auto type the given text. A timer will run to calculated the total time taken, and the length of the text divided by this time produces LDS. This will be further emphasized in the next part.

2.2.2 Selection mechanism

Selection is the process of choosing chromosomes in the whole population that can survive to the next generation. There are 3 main types of selection algorithm: roulette wheel selection, tournament selection and ranking selection:

- **Roulette wheel selection**: Every chromosome is assigned fitness resulting from the fitness function. The value is used to define the rate of it being chosen.
- **Ranking selection**: Similar to roulette wheel selection, but the difference is instead of the value from the fitness function, the ranking 1^{st}, 2^{nd}, ... It indicates that the how strong a chromosome is doesn't matter, only the ranking of those is vital.
- **Tournament**: The whole set is split into groups, each has a number of chromosomes. The top highest ranking is chosen, without any probability.

2.2.3 Crossover and mutation

Crossover is the process of regenerating another chromosomes by combining the genes of those. Mutation is when a chromosomes mutate (its genes change). Combining, this two factor will decide the strength of the next generation.

My crossover algorithm will also made up of several parameters fitness score, which is used for evaluating the probability of genes being chosen.

Crossover: Starting of with an empty keyboard, for a key position (gene) the crossover algorithm will randomly selecting each key (gene) from both keyboard (chromosome) with the probability is the fitness function. If a key (gene) is already on the keyboard (chromosome), they will be left empty and will be filled in later after the whole new keyboard (child) is produced.

Mutation: The algorithm swap the position of the key (gene) with each other. A parameter will decide how often will it be mutated and the intensity of the mutation (how many keys (genes) are swapped)

2.3 Keyboard simulation

2.3.1 Data collection

Simple text will be taken from a Random Text Generator Website by HTML scraping. The website generates dummy texts, which are just sentences with out any meaning, grammar, but just words together. There are some advantages and disadvantages, but I believe this is more suitable because the text is being randomized but still have meaning, giving higher consistency when evaluating

Moreover, source code using C++, Python or JavaScript is also used to compare how the keyboard layout changed base on different type of text. Last but not least, articles and books are used as a method of reference.

However, when being tested, keyboard are given real meaningful text to test on. This ensures both consistency and realistic for the keyboard.

2.3.2 Design

The GUI will have to consist of choices to choose, like choose what keyboard generation to run, or choose the default one like QWERTY, DVORAK or COLEMAN.

There are lots of experiment where people neglect special characters such as '?' or '!' or ' " '. However, this research will take into account of those letter. In addition, the GUI will be able to produce a letter heat map, which plays crucial in analyzing a keyboard.

3 Results

3.1 Simulation 1 - Simple Layout

In the first simulation, I only use simple text to see how the keyboard change assuming there is no special characters apart from '.'. The keyboard produced is

As shown on the image can see, I have also added the frequency of letters used to make it easier to analyze.

It is apparent that most frequent letters are placed within the home row. There are some special cases like "I" and "R" being placed in the highest row, but this is because of the charactersitic of the auto typer that I created.

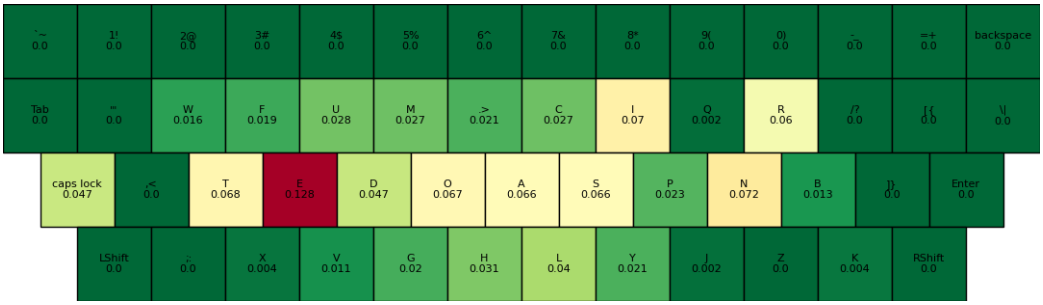

Figure 1: Simulation 1

It does not tend to return to its original position after finish typing, hence those letters' position are chosen randomly (because others column letters like "P" or "J" is not usually used.

Thus, in the middle columns, they are placed in the home row as it creates difference when the frequency of column letters "M", "H" increased. This has proved the idea of the some of the paper:

"Maximize the use of the home row"

However, the caps ",¡" is another issue. The image shows that "Caps Lock" and "Shift" is used a lots, but they cannot change their positions because of it unique design (not being a square). Therefore, the algorithm modified by reducing the pressure on pinky finger, which is also the slowest one. As a result, ",¡" is placed their.

This is what I have not found in all of the papers I have reviewed because most of the time they do not include the use of Shift and Caps Lock in the fitness function.

I have also noticed that in the column rows of "E", other letters are used much less frequently, and most of the load is on the index finger. The first part is also applied to other efficient keyboard layouts (but not mentioned), but most of them have the load distributed more regularly

Although it only appears vaguely, it is a point that I have to take into consideration.

3.2 Simulation 2 - Specialized layout

In the second simulation, I attempted to create a specialized keyboard used for coding. I used open source code Using Python, C++, JavaScript to

train my model.

Figure 2: Simulation 2

The keyboard layout is this simulation is much more chaotic, however, it is characteristic is fairly opposite to the first one. First of all, "E" is placed in the right pinky finger, which is the worst place as it has to take care of the most keys. It is also the slowest finger. Secondly the load is distributed equally on each side and quite similar to the QWERTY in how the keys are distributed.

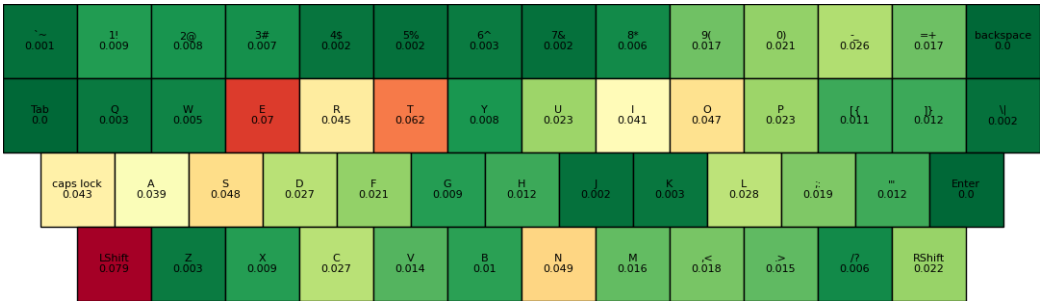

Figure 3: Simulation 1 - QWERTY layout

However, layout that are said to be efficient like DVORAK (Figure 4) or COLEMAK (Figure 5) can still keep its stability while typing code.

Figure 4: Simulation 2 - DVORAK layout

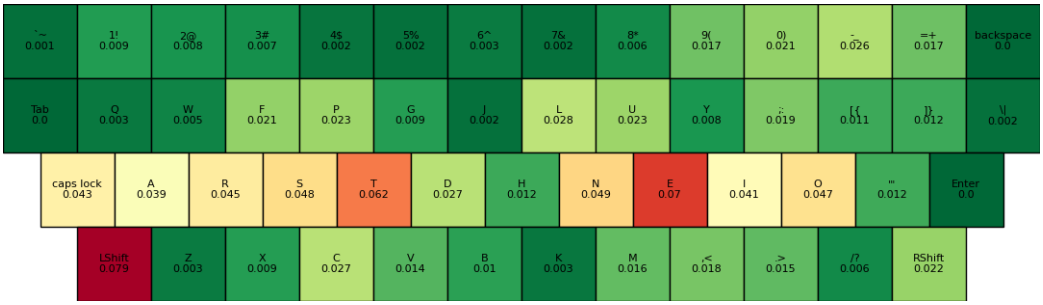

Figure 5: Simulation 2 - COLEMAK layout

The fitness score of this model varies a lots after generations, suggesting that the layout is not efficient. I believe this is because of the difference of different languages.

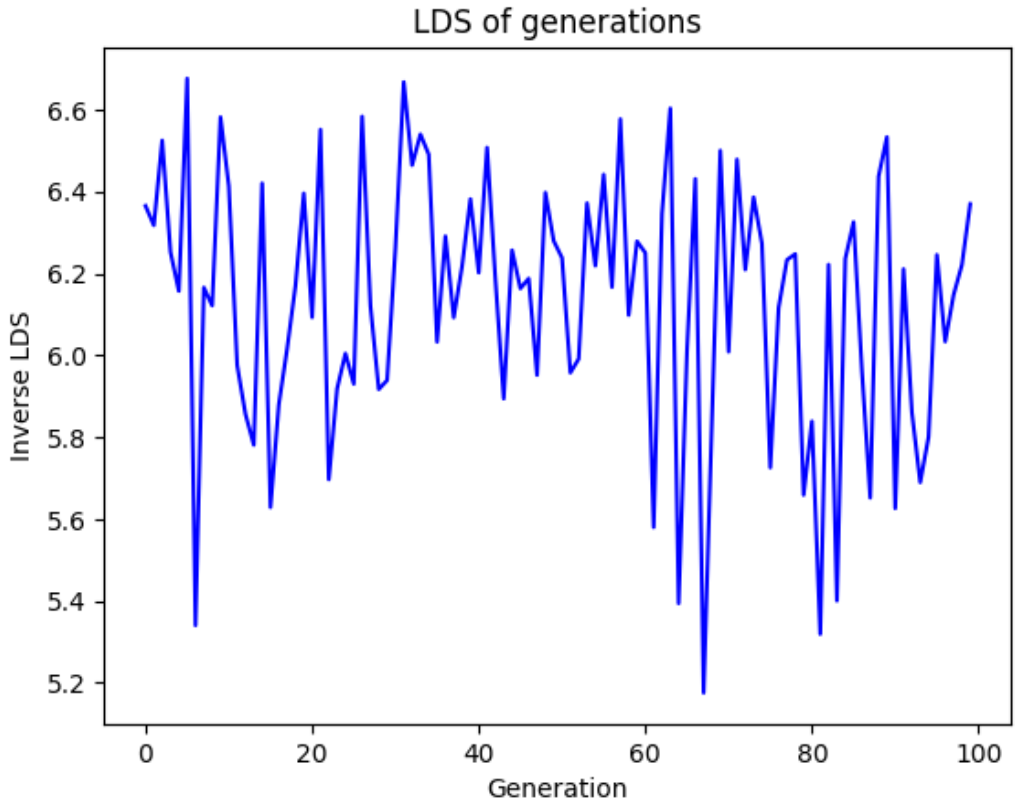

Figure 6: Simulation 2 - Fitness over generations

3.3 Simulation 3 - Combination of texts

Figure 7: Simulation 3

This keyboard meets all the requirements mentioned above

- Frequently key are placed in the home row
- The left pinky finger are responsible for Caps Lock and Shift, hence the original position is replace by a less frequently used letter
- The letters are placed so that the load on both hands is similar
- The load on the right pinky finger, which is responsible for lots of keys, are reduced to little.
- For really frequent letters like "E", the load on the column is reduced significantly. It also results in "E" is typed by the ring finger as it does not have to move much
- The load tends to be stronger the closer to the middle of the keyboard.

Another feature is that less used letter like "B" and "/?" are placed in furthest to the index finger.

The algorithm has modified based on these type of features to create the most optimal layout that.

3.4 Evaluation

Everything I have just mentioned if my generated keyboard is not as efficient as traditional one, at least for QWERTY, to some extent DVORAK and COLEMAK

Keyboard will be measured based on 2 main categories:

- Speed (Letter per second) (LPS)
- Speed (Word per minute) (WPM)

Although WPM might be a more common evaluation, I believe that LPS is more precise, and that's what I used for my fitness function

This is the result table of each testing for each keyboard layout (LPS)

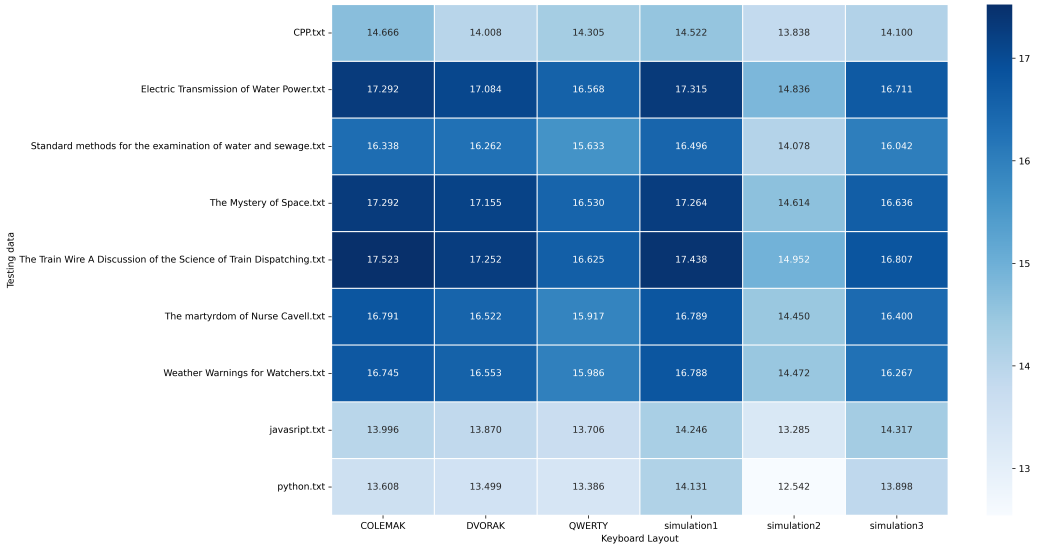

Figure 8: Evaluation of LPS

Although simulation 2 is the worst, simulation 3 performs better on the code than QWERTY.

However, the first simulation outweighs all other keyboards, even QWERTY and DVORAK. This is the overall performance of the keyboard

Evaluation	COLEMAK	DVORAK	QWERTY	simulation1	simulation 3
mean	16.027901	15.800454	15.406276	16.109685	15.686443
std	1.519030	1.546078	1.271311	1.393361	1.212922
min	13.608439	13.499283	13.386347	14.130729	13.897808
25%	14.665638	14.007975	14.304837	14.521642	14.317027
50%	16.745436	16.521763	15.916713	16.787782	16.266712
75%	17.291723	17.083872	16.530375	17.264368	16.636174
max	17.522922	17.251700	16.625212	17.437645	16.806806

Table 1: Performance of keyboard layouts (LPS)

It is clearly seen that the first simulation has the best overall performance, even higher than COLEMAK and much higher than DVORAK. Although simulation 3 generations product worse keyboard than COLEMAK and DVORAK, it performs better than QWERTY itself.

Moreover, simulation 3 has much lower standard deviation comparing to other layout, indicating more stability for typists.

4 Discussion

Despite having said that, there are are not concrete evidence that those are the best keyboard layouts, and there are many others things that need to be improve.

1. Fitness function: there should be more aspects to consider, like how the paper using ACO has used. Some aspects that I should take into consideration to make the result more reliable are "Hand alternation" and "Hand Poses". One other issue is with the number of text/length of those when calculating the overall fitness score.

2. Input data is also not quite trustworthy, especially in the programming text. For example, the tensorflow sources that I used should consists of many "T" "E", "N", "S", "O", "R" (as there are many variables are named "tensor"). Thus it is clearly seen in the simulation 2 that "T" is one of the most used character. There are many more instances but this is the most evident one.

3. Lastly, I'm not quite satisfied with the population, there could be more but the computation is much larger.

5 Conclusion

From the results and evaluation/discussion, I have drawn 4 conclusions, fundamental factors to create an efficient keyboard.

1. **The load on the home row should be maximized**

2. **Load on each side should be equal**

3. **Load should be stronger further inside until index finger and weaker towards the pinky finger**

4. **2 most frequently used letters should be in ring finger or middle finger.**

Acknowledgments

I am extremely grateful for to my supervisor, Mr Adam Depledge, for his huge support, who has proofread and give suggestions for my research paper.

I acknowledge the support from The National Mathematics and Science College, which provide resources and opportunities for this research.

References