EE2703 Assignment 5: Keyboard Optimization

Kaushik G. Iyer (EE23B135)

October 13, 2024

Abstract

In this assignment we were asked to take in a keyboard layout and some sample text. We then had to find the optimum placement of characters in this layout in order to make the distance travelled to type the given text minimum.

1 Usage

Place all the submitted files `ee23b135_kbda.py`, `ee23b135_layout.py`, `ee23b135_parsing_tools.py`, `ee23b135_simulated_annealing.py`, `ee23b135.py` and `ee23b135.ipynb` in the same directory.

Almost all the code (except for what is present in `ee23b135_simulated_annealing.py` is the same as what I submitted for assignment 4. Refer the the previous report to understand exactly what the different files do).

In order to run your custom layouts, you may create a json file (similar to the `layout.json` I submitted) or use `Layout.from_dict`) as described in a comment (Example of this is given in the report for assignment 4).

You may run either one of the files described below :)

1.1 `ee23b135.py`

Run the file, it will prompt you to give it the path to a valid json file (I have uploaded `layout.json` as an example of one). You will then be prompted to enter the path to some text file (I have uploaded `text.txt` as an example). Enter this to begin optimization.

1.2 `ee23b135.ipynb`

Simply run all the cells :).

2 Neighbour generation

Refer to `ee23b135_simulated_annealing.py` (`get_neighbour`).

I realized that having to copy data like the character map, home key map, finger group map etc. seemed very wasteful. Instead of doing this I figured I could simply define some mapping that remaps what a key means. This allows me to pass the previously stated data by reference.

In order to do this without having to change any previously written code, I introduce 2 new datastructures `OutputMappedList` and `InputMappedList`. These allow me to do this remapping step conveniently. For the InputMappedList I simply take in a mapping that is used to remap the input. A similar thing is done for the OutputMappedList :).

I can then define a neighour to a certain layout to just be the same layout but with one pair of keys swapped :).

3 Simulated annealing

Refer to `ee23b135_simulated_annealing.py` (`optimize`). You may change the parameters taken (like the amount of iterations done, the inital temperate and the cooling factor as you desire).

In this function I take a random neighbour of the current layout. I calculate the distance travelled in this layout. If this is better than the current layout I have I use this for further iterations.

In order to bring in some amount of randomness (To avoid getting stuck in a high minima) I also use a worse layout with some chance (inversly proportional to how much worse it is than the current and directly proportional to the current temperature).

After all iterations are done, I simply return the most optimum layout for in all of the steps taken till now :)

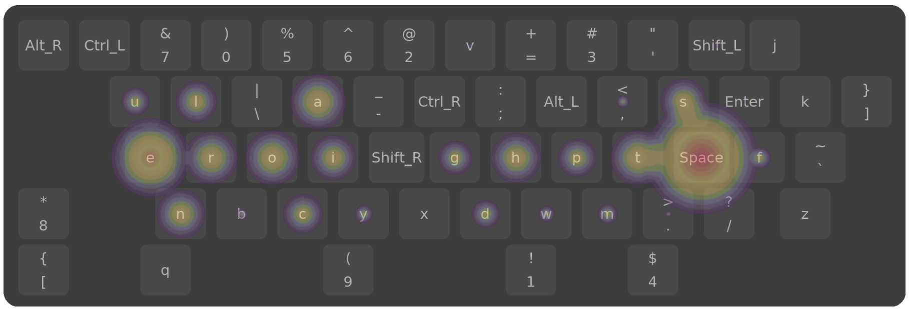

Figure 1: Optimized keyboard for text.txt

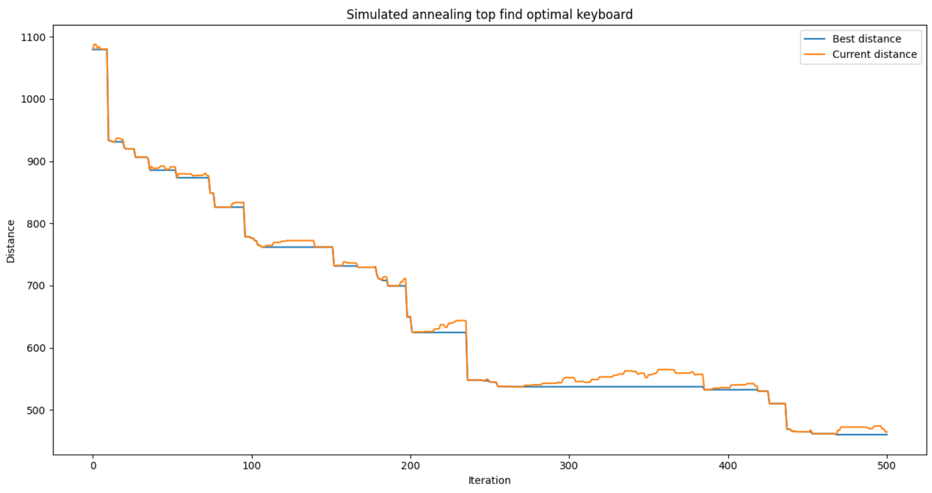

Figure 2: A graph representing the steps taken while doing simulated annealing

4 Some comments and observations

Since our model for typing distance is very simple, it is easy to guess that the best model (with allowing for moving of keys around) will be one that is bunched up around the home keys. (The qwerty layout does that decently well but there are some that do it better).

We could also make a nice initial approximation by just placing the more frequently used keys close to the home row, with less frequent ones placed further. This will always ensure that the keyboard is decently optimized :)