UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Improved keyboard layout for users with movement disorders

Author:
Zenovii Popeniuk

Supervisor:
Mr. Oleg FARENYUK

A thesis submitted in fulfillment of the requirements
for the degree of Bachelor of Science
in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2021

Declaration of Authorship

I, Zenovii Popeniuk, declare that this thesis titled, "Improved keyboard layout for users with movement disorders" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"We need to make every single thing accessible to every single person with a disability."

Stevie Wonder

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Improved keyboard layout for users with movement disorders

by Zenovii Popeniuk

Abstract

In this paper, I will describe the implementation of an improved keyboard layout. With it, the user will be able to set up the keyboard layout as user wants. I will describe used technologies, the architecture and the implementation of the program, and alternative solutions. Also, I will cover AI autocorrection and AI autocompletion topics...

Code could be found here: *Improved keyboard layout github*

Acknowledgements

First of all I want to thank my supervisor Oleg Farenyuk and Galyna Butovych for mentoring me and for all great knowledge I gathered over the years. ...

Contents

List of Figures

List of Tables

List of Abbreviations

ES	**E**ssential **T**remor
GUIs	**G**raphical **U**ser **I**nterfaces
OS	**O**perating **S**ystem
PyQt5	**P**ython **Qt** 5
PC	**P**ersonal **C**omputer
AI	**A**rtificial **I**ntelligence
DL	**D**eep **L**earning
seq2seq	**S**equence to **S**equence
CNN	**C**onvolutional **N**eural **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
scRNN	**S**emi-character **R**ecurrent **N**eural **N**etwork

Chapter 1

Introduction

1.1 Context

Last year, I with *Yevhenii Molodtsov*, *Mariya Hirna*, *Danylo - Ivan Kolinko* and *Nazariy Bachynsky* took part in Microsoft The Next AI Guardians. We took up this tournament with the concept of the Smart Keyboard, which helps users customize the keyboard as they need. Participating in this competition, we made a Proof of Concept keyboard remap program. But then we focused on AI models for text correction and completion and a business model for our program. Unfortunately, the program contained many bugs; the AI model runs on the local server. We did not implement many ideas.

1.2 Motivation

We have friends with cerebral palsy and essential tremors who suffer from typing. Every day they spend much time entering the text correctly. So that we want to learn how people with movement disorders deal with keyboard input, we need this to better understand their pain we can remove and the most universal and convenient way to do this. This research was done in the scope of the tournament.We conducted more in-depth research with each and identified the following insights:

- *"It can be challenging to reach the top row of keys."*
- *"I make many misclicks. I hit keys near to the necessary key."*
- *"It is hard to hit the key. However, if I hit, there are no problems with pressing and holding."*
- *"I learned to type on a screen keyboard using a mouse."*

1.3 Problem

Interaction with the computer is an essential part of modern life. Despite the widespread use of touch screens and speech-to-text technologies, keyboards are the fundamental input device for many users. However, there are users with movement disabilities who find it difficult to type on the keyboard. According to previous research, more than 280 million people globally suffer from Essential Tremor(*Overview of essential tremor*). Unfortunately, I have not found how many of them are active PC users, but since their illness does not affect their desire to use a computer, I estimate at least

14 million. The most challenging task for them is to hit the needed keys on the keyboard. For example, when the user wants to click the “T”, “G”, “R”, “Y”, “5” or “6” key could be misclicked.

1.4 Solution

I suggest a method to reduce the number of keys with which users need to interact. I provide full-fledged access to keyboard capabilities. The method is based on keyboard macros encoding required input sequences by the combinations convenient for each specific user. The typical macrouses repeat presses of some key to input it or any nearby key. The corresponding software package was developed implementing intelligence keys remapping and on-screen hints to the users during typing. Also, each user could create a new hotkey or remap existing ones. I also want to implement an auto-change of remapping following the active session of the program. This auto-change would allow the user to create a more unified list of hotkeys. Also, we developed AI-based auto-completion and auto-correction tailored for specific users as part of the package.

1.5 Goal

The main goal of this work is to make a fully customizable application, which allows user remapping for maximum flexibility. Since I work with data entry, the program must have minimal divisions between remap processing. I also want to provide users with ai correction and completion in real-time. The program should have the ability to set up from UI and JSON file script. In addition, it should be an open-source project written with python to maximize the ability each user adds needed for his functionality.

Chapter 2

Technologies

2.1 Qt

Qt is a cross-platform widget toolkit for creating graphical user interfaces(GUIs). The programmers could create a new application with it.

2.1.1 Main Qt Modules

- Qt Core
  It contains tools for object communication, different qt object properties constants, and trees of hierarchy.
- Qt GUI
  It provides classes for 2D graphics, buttons, images, movies, text, events handling and other. This classes could be used for setup widgets from Qt Widgets.
- Qt Widgets
  Provides classes for creating application GUI: layouts, Labels, different buttons, checkboxes and others .

It is also important to mention a powerful tool - Qt designer, with which developers can easily and quickly generate the GUI they need. For example, they could easily create a new button and then bind its click event to C++ functions or default one like an exit from the application. Similarly, many things can be implemented without code, such as changing the color when a key is pressed. Programmers could see what they have and what they would have in real-time.

2.2 PyQt5

PyQt5 is a wrapper over cross-platform C ++ Qt version 5. It provides alternative application development for Qt. Also, it includes many library bindings of original Qt. With it, I have written remap and AI hints for users.

2.2.1 Used PyQt5 technologies

I used the next functions of this library:

- QApplication
  class that manages GUI and main application settings

- QMainWindow
  class that manages main application window and its settings

- pyqtSignal, pyqtSlot, QObject
  This could be used to implement communication between different objects and threads.
  pyqtSignal - defines new qt signal.
  pyqtSlot - decorates a python method that could be connected to pyqtSignal.
  QObject - base Qt object.

- QHBoxLayout, QVBoxLayout
  Class to construct horizontal or vertical box layout objects.

- QMovie
  class for playing movies

2.3 PyTorch

Pytorch is a free, cross-platform and open-source machine learning library. Most of the deep learning models are running on it. This library allows users to run models on CPU and GPU. With GPU, it speedup model training and running very much. We use this library for running our own correction words model. We check if GPU is available, and then if yes, then we run the model on it. If not, then we run model on CPU.

2.4 Python Keyboard Library

Python keyboard library is a cross-platform library that allows users to control the keyboard fully. I created few approaches that allow me to remap keys, show hints for remapped keys, hook keys, simulate key events, write information to buffer, and create new shorthands. Under the hood, this library uses the Win32 API library through the ctypes module. It is using the windows kernel function to create proper events and a proper hook for these events.

2.4.1 Used Python Keyboard technologies

I used the next functions of this library:

- keyboard.hook_key(key, callback, suppress=False)
  This function hook key to Callback. If suppress is True, then another program would receive this key when it would be pressed.

- keyboard.write(text, delay=0, restore_state_after=True, exact=None)
  Simulates text input to OS.

- keyboard.send(hotkey, do_press=True, do_release=True)
  Simulates hotkey send. If do_press False, then send would not generate press event. If do_release is False, then send would not generates release event.

- keyboard.wait(hotkey=None, suppress=False, trigger_on_release=False)
  It waits until hotkey would be pressed. If suppress is True, then another program would receive this key when it would be pressed. If trigger_on_release is True, it will wait until hotkey would be realized.

- keyboard.add_hotkey(hotkey, callback, args=(), suppress=False, timeout=1, trigger_on_release=False)
  Callback would be invoked when hotkey would be pressed. If suppress is True, then another program would receive this key when it would be pressed. If trigger_on_release is True, it will wait until hotkey would be realized.

- keyboard.add_abbreviation(source_text, replacement_text, match_suffix=False, timeout=2)
  add new abbreviation. When the user input source text and press space, then source text would be changed by replacement text

- keyboard.unhook_all
  Unhook all new events.

Chapter 3

Solution Overview

I have implemented an application for remapping keys. This application reads the JSON file and configures the remap relative to the settings.

3.1 Configuration Example File

I created configuration.json architecture to allow the user to create his unique keyboard setup:

```
{
  "layouts": [
    {
      "name": "Layout 1",
      "shortcuts": {
        // This is described in the Button Remap section.
      },
      "shorthands": {
        // This is described in the Shorthands section.
      },
      "key_remap_configuration": {
        "key_remap_list": {
        // This is described in the Shortcuts Remap section
        },
        "max_time_delay_for_multiple_press": 1,
        "key_remap_time_delay_for_long_press": 1,
        "hints": true
      },
      "ai": {
        "correction": true,
        "completion": true
      }
    },
    {
      "name": "Layout 2",
      "activate_program": "chrome.exe"
    }
  ]
}
```

Users could create few layouts for the keyboard to make it more flexible for different programs. For example, there is another layout for chrome; and there is another layout for Visual Studio Code. This solves the problem when the two programs have different shortcuts for one command. Switching between these layouts could be done using shortcuts or by assigning a program to activate the layout.

3.1.1 Loading Application Hint

Since the improved keyboard layout takes time to launch, I have added a loading hint so that the user understands whether the application is running or not.

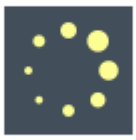

FIGURE 3.1: Loading Application Indicator

Loading Application Hint Implementation

For this implementation, I created QLabel and QMovie to display loading.gif. After that, we started gif animation. For QLabel, I set the property that this label is always on top and the window is frameless so that users only see the gif without application window.

3.2 Keys Remap

3.2.1 Configuration file

```
"key_remap_configuration": {
    "key_remap_list": {
      "a": [
        "a",
        "h",
        "f",
        "g"
      ],
      "z": [
        "z",
        "r",
        "e",
        "f"
      ]
    },
    "max_time_delay_for_multiple_press": 1,
    "key_remap_time_delay_for_long_press": 1,
    "hints": true
  }
```

3.2.2 Overview

With key remap, the user could remap any button to any other button so that there would be few keys on one. Let us suppose that the user remapped "a", "h", "f" and "g" too alike in the configuration file, then when "a" would-be pressed, on the screen would be a hint with remapped button if hints are enabled. If "a" would be pressed two times, "a" would be changed on "h", if three times, then "f" and so on(See proper hint image). This method is first. The second is to press key "a" and wait until it would be changed to need character. The hint would be the same.

max_time_delay_for_multiple_press - is the maximum time delay between key presses on the same button to change the active button to next.

key_remap_time_delay_for_long_press - is the minimum time delay between key clamping to change button to next.

Hints - if true, then it shows hints. If no, then hint would not be shown.

3.2.3 Implementation

```
button = KeyButton(key, keymap["key_remap_list"][key],
                   max_time_delay_for_multiple_press,
                   key_remap_time_delay_for_long_press)
keyboard.hook_key(key, button.hook_press, True)
```

With keyboard.hook_key I hooked remapped keys with KeyButton. KeyButton is a class that is responsible for the logic of the key remapping. When the key is pressed, the event would be generated and sent to button.hook_press. Then KeyButton would check if there is a need to activate the next character. If yes, then the next character would be activated. For activate, button KeyButton sends backspace and new character as a keyboard event. The hint would show a new current. Implementation:

```
keyboard.send('backspace')
keyboard.send(self.keys[self.current_key])
self.hint.activate_button(self.current_key)

class KeyHint:
    ...
    def activate_button(self, index):
        self.buttons[self.current_button].setStyleSheet(
            self.nonActiveButton)
        self.current_button = index
        self.buttons[self.current_button].setStyleSheet(
            self.activeButton)
        self.current_key = (self.current_key + 1) % len(
            self.keys)
```

3.2.4 Hints

FIGURE 3.2: Key Remap Hint "a" active

FIGURE 3.3: Key Remap Hint "h" active

FIGURE 3.4: Key Remap Hint "f" active

3.3 Shorcuts Remap

```
"shortcuts": {
    "ctrl+insert": {
      "mode": "shortcut",
      "value": "ctrl+c"
    },
    "shift+insert": {
      "mode": "shortcut",
      "value": "alt+c"
    },
    "alt+w": {
      "mode": "shorthand",
      "value": "Hello world"
    },
    "ctrl+2": {
      "mode": "activate_layout",
      "value": "Layout 2"
    },
    "ctrl+1": {
      "mode": "run_program",
      "value": "chrome.exe"
    },
    "ctrl+5": {
      "mode": "insert_suggestion_word",
      "value": ""
    },
    "ctrl+i": {
      "mode": "insert_suggestion_sentence",
      "value": ""
    },
    "ctrl+4": {
      "mode": "clear_ai_sentence",
      "value": ""
    },
    "ctrl+3": {
      "mode": "clear_ai_word",
      "value": ""
    }
}
```

To implement it I used proper lambda function and keyboard.hook_key. But for different modes there are different commands:

- shortcut mode
  Simple shortcuts remap

- shorthand mode
  Simple insert custom user string

- run_program mode
  Run program with name equel value

- activate_layout mode
  Activate another keyboard layout

- insert_suggestion_word mode
  Insert suggestion ai complete word

- insert_suggestion_sentence mode
  Insert suggestion ai corrector sentence

- clear_suggestion_word mode
  Clear suggestion word to prevent false completion

- clear_suggestion_sentence mode
  Clear suggestion ai complete sentence to prevent false correction

3.4 Shorthands

```
0 {
1   "shorthands": {
2         "@@": "zenovij.popenyuk@gmail.com",
3         "_phone_": "phone example"
4       }
5 }
```

This is the easiest part, as it is essentially just using add_abbreviations from the keyboard library. If the user inputs some shorthand and then presses the space button, then a whole phrase would be inserted. Implementation:

```
0 keyboard.add_abbreviation(abbreviation, abbreviations[
    abbreviation])
```

Chapter 4

AI Correction and Completion

4.0.1 Configuration file

```
{
    "ai": {
        "correction": true,
        "completion": true
      }
}
```

4.0.2 Overview

Danylo - Ivan Kolinko developed few deep learning(DL) models for text correction problem:

- semi-character RNN classification
- semi-character RNN regression
- character CNN
- seq2seq approach

We tested these models and found that semi-character RNN with words classification is the best for us, as it has the best accuracy and does not use many computer resources during keyboard input. So that we would describe only this model, but *Github: Comparing spell correction approaches* contains more information about other models.

4.0.3 Dataset

Our problem is relatively narrow, so it became clear that it is necessary to create the dataset containing random errors and user misclicks. For this purpose, *Danylo - Ivan Kolinko* used books from *Project Gutenberg* as the base dataset for generating a target dataset with KeyboardAugmenter, which simulates user misclicks.

4.0.4 Inputs

Our model accepts vectors that consist of three sub-vectors embeddings: two vectors with one-hot embeddings of the first and last characters in the word. The last subvector is the count of each character in the middle.

4.0.5 Semi-character RNN Model

The scRNN was initially introduced in the *Robsut Wrod Reocginiton via Semi-Character Recurrent Neural Network*. The authors of the original paper chosen two stacked LSTMs with a fully connected layer on top. Softmax and CrossEntropy is used as a loss function since the problem formulated as classification word to word. *Danylo - Ivan Kolinko* created a different method, which is to use as a target, not the word itself but its embedding in the k dimensional space.

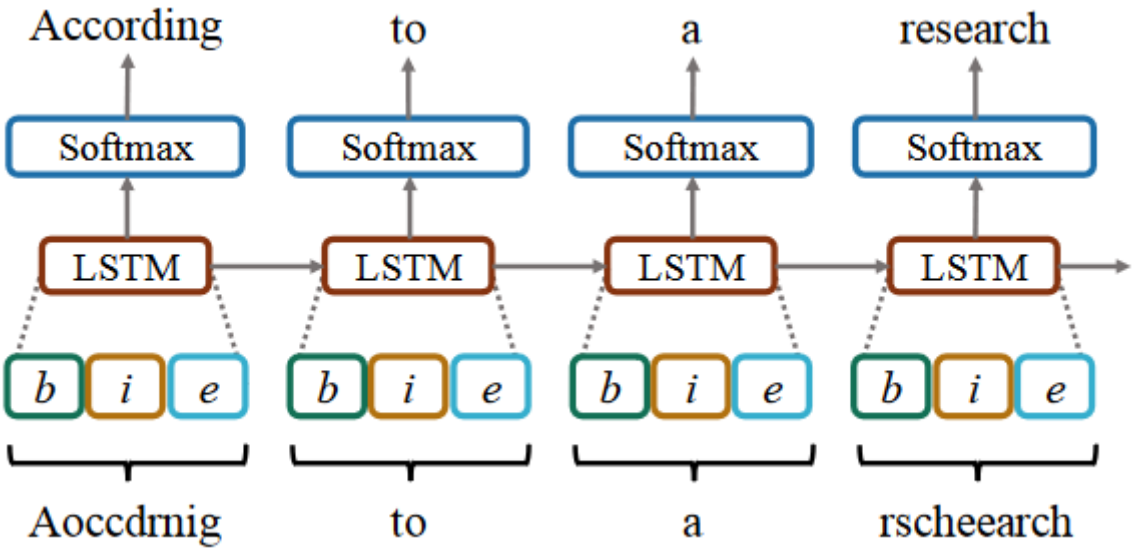

FIGURE 4.1: Semi-character RNN Model

4.0.6 Training

Semi-character RNN was trained with a target dataset where 40 percents of the words were contaminated. Some characters had either been removed in these sentences, some new characters have been added or interchanged.

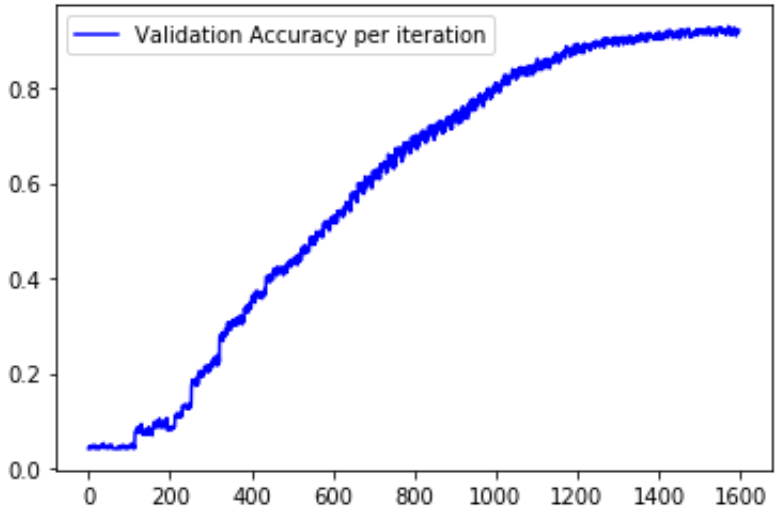

FIGURE 4.2: Semi-character RNN Model accuracy

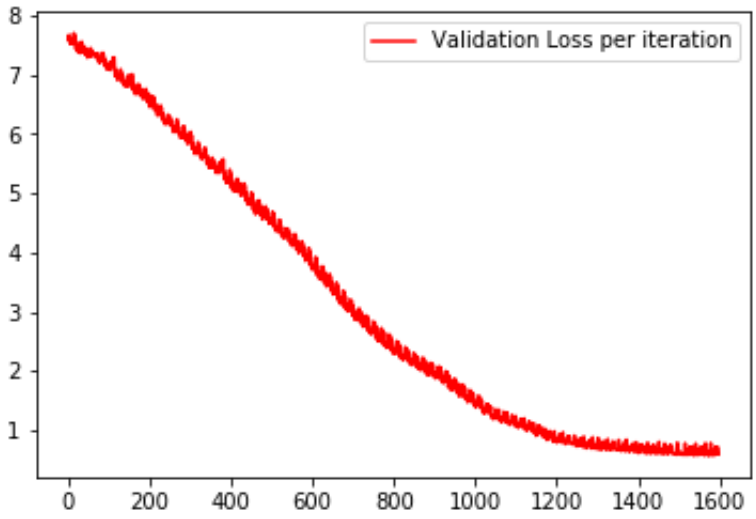

FIGURE 4.3: Semi-character RNN Model losses

4.0.7 Conclusions

We decided that the scRNN classification performs the best and shows an accuracy of 92.7. We used it as the primary model that corrects user input data.

4.0.8 Usage implementation

AI Completion Hints

If the user includes ai completion, the application starts to listen to each user character input. He adds new letters and tries to predict the current word.

```
0  def start_listen(self):
1      for letter in get_default_keyboard_layout():
2          keyboard.hook_key(letter, self.on_press, False)
3
4      keyboard.hook_key('backspace', self.on_backspace, False
           )
5      keyboard.hook_key('space', self.on_space, False)
6
7      for character in get_stop_characters():
8          keyboard.hook_key(character, self.
               on_stop_character_press, False)
```

When the user clicks, character AI Supporter sends this character to the scRNN and gets the predicted word from the model.

```
0  def add_letter_and_predict(self, letter):
1      self.last_word += letter
2      return self.predict(self.last_word)
```

When the user clicks, character AI Supporter sends this character to the scRNN and gets the predicted word from the model.

```
0  def add_letter_and_predict(self, letter):
```

```
    self.last_word += letter
    return self.predict(self.last_word)
```

If the user presses a specific shortcut, then the predicted world would be inserted. b is backspace.

```
keyboard.write('\b' * (len(self.word)+1) + self.suggestion
   + " ")
```

FIGURE 4.4: Button Remap Hint

AI Correction Hints

The same mechanism is used for ai corrector, but AI Supporter just saves inputted characters by the user during user input. And then, when the user presses a specific shortcut, ut sends a sentence to model and get corrected. Then it blocks all input and waits until the user would choose to need words. The user could change with 'up' and 'down buttons. After that, when user would press 'esc' the chosen worlds would be inserted.

```
// add new character to sentence
if self.is_correction:
    self.sentence += self.current_character
```

```
def insert_suggestion_sentence(self):
    // correct self.sentence
    words = re.findall(r'\w+', self.sentence)
    words_with_punct = re.findall(r"[\w']+|[.,!?;]", self.
       sentence)

    // correct words with model
    corrected_sentence = self.corrector.correct(words)
    corrected_words = corrected_sentence.split(' ')

...
// show user hints
chosen = setup_hint(self.correction_hint, to_show_hint)
```

```
// show hints and get right corrected words
def setup_hint(keyboard_hints, hints):
    // show hints
    keyboard_hints.set_hints(hints)
```

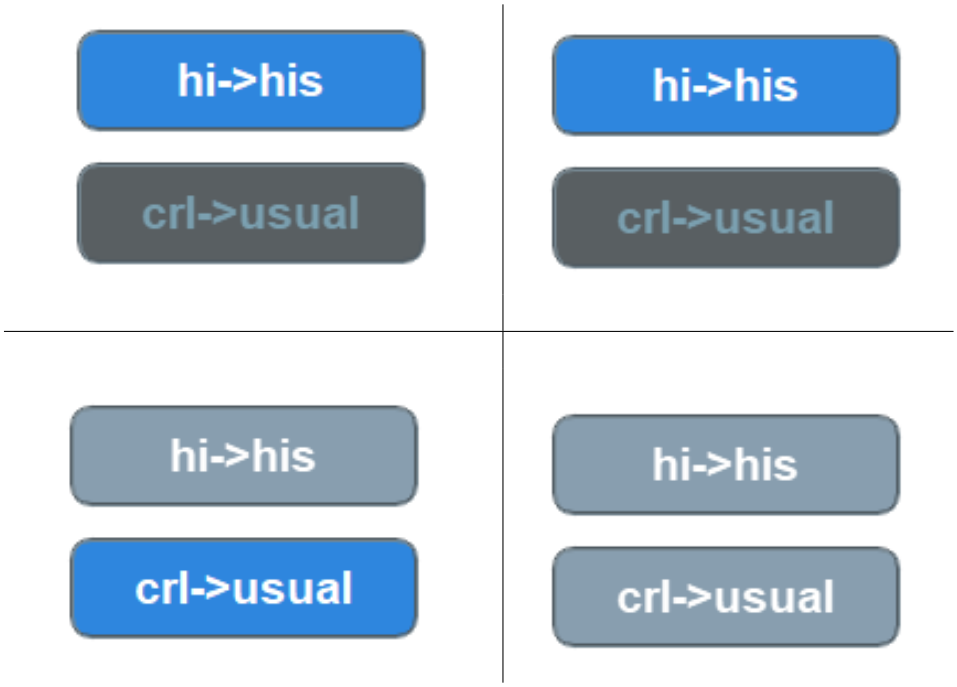

TABLE 4.1: AI Correction Hints

```
    keyboard_hints.window_show()
    // observe needed buttons
    key_down = keyboard.hook_key("down", keyboard_hints.
       activate_next_button, suppress=True)
    key_up = keyboard.hook_key("up", keyboard_hints.
       activate_prev_button, suppress=True)
    key_enter = keyboard.hook_key("enter", keyboard_hints.
       set_active, suppress=True)

    // wait for escape key
    keyboard.wait("esc", suppress=True)
    keyboard_hints.window_hide()
    keyboard_hints.choosenButtons.clear()

    // unhook unneeded keys
    keyboard.unhook(key_down)
    keyboard.unhook(key_up)
    keyboard.unhook(key_enter)

    // return right corrected words
    return keyboard_hints.get_active_index()
```

```
keyboard.write('\b' * (len(self.sentence)+1) + new_sentence
    + " ")
```

Chapter 5

Application architecture

5.1 Overview

As previously described, there are two applications, one - improved keyboard layout manager, and the other just improved keyboard layout.

5.2 Patterns

Two patterns were used for this project: model view controller and observer. Model view controller as it is standard for Qt and observer in order to subscribe to various events.

5.3 Improved Keyboard Layout Components

CorrectionHint - class that represents correction hint windows that are shown to the user correctly. In init, it initializes all needed objects and settings. Also, it contains functions for a style change, activation of the previous or next button, and activation current button.

KeyboardController - class that handles json configuration file and setup all needed objects and others. It have few functions:

- set_layout - set layout by index
- set_layout_by_name - set layout by name
- __clear_keyboard_configuration - clear current configuration
- __install_keyboard_configuration - install new keyboard configuration

KeyboardHintController - a file that contains a function that calls to show correction hint to the user.

KeyboardLayoutUtilsFunctions - Utils functions for keyboard layout.

KeyButton - class for multiple key remap.

KeyHint - class for showing hint for key remap.

PredictionHint - class that shows hint for auto-completion model.

RemapController - a file that contains functions that set up remapped buttons.

ShortcutsController - file that contains functions that setup shortcuts buttons.

Styles - class for applicaiton style.

dataloader - class for loading data for AI.

AICorrectorModel - class for model implementation.

AISupporter - the class that hooks all needed keys, saves user input text, then sends it to scRNN and shows user proposed solution.

Chapter 6

Alternative Solutions

6.1 Logitech Craft

It is a premium wireless membrane keyboard. It adapts to what users are making. It allows the user to remap shortcuts and make him create a new one.

Pros:

- No delay as it is a hardware device
- It has its own application for setting up different special commands
- It haves many different remap modes for running new application, close application, pause, start movie, and others
- It has its own hardware slider that helps designers

Cons:

- It cost 200 $
- No key remap
- No full shortcuts remap
- No different keyboard layouts

6.2 PowerToys

It is a Microsoft open-source program that allows users many new things, such as keys remap, autocomplete, and new window layout(not new keyboard layout.)

Pros:

- It allows the user to create a new windows layout (Fancy Zones)
- It has single key remap
- It has single key shortcuts remap
- It has a color picker
- It has nice GUI

Cons:

- No multi-key remap

- No multi-key shortcut remap
- No user hints in real-time
- No different keyboard layouts
- It is a preview version, so that there many bugs

6.3 KeyTweak

Pros:

- It has single key remap
- It has three different offers method

Cons:

- Has bad GUI
- No multi-key remap
- No multi-key shortcut remap
- No user hints in real-time
- No different keyboard layouts
- It is a preview version, so that there many bugs

6.4 Microsoft Keyboard Layout Creator

Pros:

- It allows the user to create a few keyboard layouts for different languages and keyboards type.
- It has single key remap
- It allows the user to create a new registry for characters input
- It has simple GUI

Cons:

- No multi-key remap
- No multi-key shortcut remap
- No user hints in real-time
- No different keyboard layouts

6.5 AutoHotkey

Pros:

- It allows the user to remap keys with different approaches
- It allows to create custom executable scripts

- It allows to create new key binds, runs executable
- It has simple GUI

Cons:

- Need much time to understand the app
- It is not user friendly
- No user hints in real-time
- No different keyboard layouts

Chapter 7

Testing

7.1 Testing usage of computer resources

In order to determine if I solved the problems of peoples with ET, I did user testing. The key insights I got:

- Users with ET were confused with advanced settings. They were afraid to do something wrong.
- Users get used to remapping pretty fast.
- Key remap hits is awesome
- It took some time to learn shortcuts for extended functionality (shorthands, AI). However, as far as they learn them, as users mentioned, that functionality makes many things easier.
- It seems the functionality is quite easy to understand and use. However, the first times of using a program are painful for most users.
- I need to add more hints and maybe some kind of tutorial on what all settings mean and how to use them.
- One large help on the shortcut is missing so that absolutely all settings for the current layout are shown.
- Improved keyboard layout manager contains a lot of bugs that I need to fix. Also, advanced settings should be added here.

7.2 Testing usage of computer resources

It is also important to test how many computer resources the program uses. Testing resourcing using was easy, but to check the delay of input text with the improved keyboard layout, I created a separate algorithm, which ten times ran the test and wrote the data to separate folders of each test. This data was pictures. My algorithm launched the application, start video recording, and then pressed the button. When the algorithm receives an event that button is pressed, it divides the video into few screenshots, each 25 ms.

Computer	Acer Nitro 5 AN515-44-R9YH	Asus Vivobook x556uq
CPU	AMD Ryzen 7 4800H	Intel i5 7200U
Memory	16 Gb dual slot	16 Gb dual slot
Memory speed	3200 MHz	2133 MHz
GPU	Nvidia GTX 1650Ti	NVIDIA GeForce 940MX
CPU Usage without AI and hints	2.1% - 6.4%	5.9% - 8.7%
GPU Usage without AI and hints	0/4 Gb	0/2 Gb
CPU usage with AI and hints	3.8% - 5.8%	7.7% - 9.3%
GPU usage with AI and hints	1.1/4 Gb	1.3/2 Gb
Key remap delay without hints	< 25 ms	< 50 ms
Key remap delay with hints	< 25 ms	< 50 ms
Shorcuts remap delay without hints	< 50 ms	< 50 ms
Shorcuts remap delay with hints	< 75 ms	< 100 ms
Ai correction delay	< 125 ms	< 350 ms
Ai completion delay	< 300 ms	< 500 ms

TABLE 7.1: Testing usage of computer resources

Chapter 8

Conclusion

I implemented a program that allows users with disabilities to customize keyboards. Users can enter everything with just a few keys. As comparisons with alternatives have shown, my app has many features that no competitors have. The main features of the application:

- It allows multi-key remap
- It has multi-key shortcut remap
- It has user hints in real-time
- It has different keyboard layouts
- It has simple hints for remapped keys and ai
- It has ai correction and completion

Testing with users showed that assumptions were correct. Also, this approach did not use many computer resources.

Bibliography

Danylo - Ivan Kolinko. URL: https://ua.linkedin.com/in/danylo-ivan-kolinko-208b58201?trk=public_profile_browsemap_profile-result-card_result-card_full-click.

Improved keyboard layout github. URL: https://github.com/popenyuk/Improved-keyboard-layout.

Kolinko, Danylo. *Github: Comparing spell correction approaches*. URL: https://github.com/Kolinko-Danylo/DL-approaches-for-spell-correction.

Mariya Hirna. URL: https://www.linkedin.com/in/mariya-hirna-10a555153.

Nazariy Bachynsky. URL: https://www.linkedin.com/in/bachynskyn/.

Project Gutenberg. URL: https://www.gutenberg.org/.

PyQt5. URL: https://pypi.org/project/PyQt5/.

Python keyboard library. URL: https://pypi.org/project/keyboard/.

Pytorch. URL: https://en.wikipedia.org/wiki/PyTorch.

Qt. URL: https://www.qt.io/.

Sakaguchi, Keisuke et al. *Robsut Wrod Reocginiton via Semi-Character Recurrent Neural Network*. URL: https://arxiv.org/pdf/1608.02214v2.pdf. 2017.

Yevhenii Molodtsov. URL: https://ua.linkedin.com/in/moyevhenii.

Zesiewicz, Theresa A et al. *Overview of essential tremor*. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2938289/#:~:text=The%20age%2D%20and%20gender%2Dadjusted,incidence%20of%2023.7%20per%20100%2C000.&text=The%20incidence%20of%20ET%20rises,disease%20can%20also%20affect%20children.&text=Approximately%204%25%20of%20adults%2040,older%20are%20affected%20by%20ET.. (07 September 2010).