This documentation has been written by Michael Meckl, the demo video has been recorded by Johannes Lorper.

Goal

The goal of this study was to compare two keyboard text input techniques regarding the speed with which a user can enter a given text (a standard “text copy task”). Therefore, a standard, unenhanced keyboard input was compared with our own input technique which enhanced the keyboard input with autocompletions shown as a dropdown list. Our alternative hypotheses was “Participants that use our enhanced keyboard input with autocompletion can enter text faster than with unenhanced keyboard input” while our null hypothesis stated that there was no difference between both conditions or using autocompletion would even lead to slower text input. The application was created with Qt Designer [1] and implemented in Python with the PyQt-Framework [2].

Experiment Design

We used a within-subjects design with four conditions. The conditions were counterbalanced for each participant with a balanced latin square.

At the beginning of the experiment users were shown a short introduction with an explanation of the upcoming tasks and were asked to make sure that they were in an undisturbed place for the time of this study so they would be able to fully concentrate on the task. As soon as they clicked on the button at the bottom, the actual trial with the four conditions for this participant started. The procedure was the same for every condition: first, users were shown their current task (i.e. if there would be autocompletion or not) as well as the instruction to enter the text as fast and error-free as possible but ignore any errors that were made. Below the task description was a short example text and an input field where they could test the current input technique to get familiar with it or simply to adjust from the previous condition. These short example pages also gave the participant time to relax between tasks and thereby prevent fatique effects. The actual task started on the next page. The instructions and the current task description were shown again at the top as a reference and below them appeared the text that they were supposed to enter in a text field at the bottom. Time measuring started at the moment the first character was entered in the text field and ended when the last character of the last sentence had been entered. Measuring was simply done with the `time.time()` function in python to get the current timestamp. A word was considered to be finished when the current input key was one of: `, ; : . ! ?` as these resemble typical separators. If the key was one of `.!?` a sentence was counted as finished as well. Additional sanity checks were implemented to prevent some errors, like ignoring a whitespace after one of these separators so it won’t count twice. After a sentence has been finished, the timer for the next sentence was immediately restarted while the timer for the next word was only restarted when the user entered a word character (`\w`) and the last character before wasn’t a word character or a digit (checked with `last_char.isalnum()`).

During these tasks each key press, entered word and sentence was logged with start and end timestamps. After each task the next condition according to the balanced latin square was presented to the participant in the same way. After the last condition the participant had to fill out a short questionnaire containing demographic questions like age, gender and occupation as well as two questions about the participant’s frequency of keyboard usage and a subjective estimation of his own input speed to to find out more about the users’ experience level concerning keyboard usage and text entry.

The texts used for the task in this study have been taken from https://www.blindtextgenerator.de/ (Werther) and are parts of a short extract of "Die Leiden des jungen Werther" from Johann Wolfgang von Goethe. It was chosen because it is a meaningful text unlike other blind texts like 'Lorem Ipsum' but on the other hand shouldn't be too easy for the participants. Both parts (text 1 and 2) consist of 58 words. The example texts were taken from the same page but from other blind texts to prevent too many similarities with the real experiment text. The first one was taken from "Er hörte leise" and the second example text was taken from "Kafka". Both consist of 19 words.

Our dependent variable was the speed with which the user entered specific words and sentences. Our independent variables were the combinations of input technique and task text:

C1: unenhanced input + text 1

C2: unenhanced input + text 2

C3: input with autocompletion + text 1

C4: input with autocompletion + text 2

Two different texts were chosen to migitate learning effects. A confounding variable might be the varying difficulty of these texts but these shouldn't cause a problem because of the counterbalancing applied to every trial.

Custom Input Technique

We enhanced the the keyboard input with an autocomplete functionality provided by the *QCompleter* Class from Qt [3]. Possible autocompletions were shown as small dropdown list (limited to 3 entries each for clarity) after at least 3 characters had been entered.

We chose to let the user select one of these three options by pressing the 1, 2 or 3 on the keyboard (the topmost item would therefore be selectable by pressing '1') instead of selecting them with arrow keys and the enter key so it will be faster as the arrow keys are generally further away from the regular finger positions when entering text than the number keys at the top of the keyboard. To prevent users from using the enter key despite the instruction to use 1, 2 or 3 we blocked the key press event of the enter key in the PyQt application. This way we could control that every participant used the same techniques when interacting with the autocompletion. The stylesheet for the dropdown elements has also been adjusted so it won't show any highlights which might tempt an user to select this element with the enter key even though this is not possible.

The QCompleter needs a corpus or any other model of strings to generate autocompletion results. As our texts were in German, a German corpus was needed as well. For this, we use the TIGER corpus hosted by the University of Stuttgart, which consists of words from the german newspaper "Frankfurter Rundschau" [4].

Participants

The study was conducted by 4 participants ranging in age from 22 to 27 years (m = 24,5, sd = 2,5). All participants were male and studied media informatics at the University of Regensburg.

Limitations

A huge limitation of this study is the fact that errors and therefore the input accuracy have not been logged as well which makes the results of the evaluation questionable at best. Faster input speeds might simply be due to less accurate inputs, i.e. at the expense of a lot more errors.

Another confounding variable in our design is the fact that participants were asked not to correct any mistakes they make but this wasn't controlled for. However, logs show that backspace was used quite often, maybe even unintentionally because participants might be used to correct their mistakes. This makes the results of these participants less comparable to participants who actually didn't correct anything. Logging errors and accuracy would be one way to migitate this limitation. Other options could be implemented in the code directly: for example, the use of backspace or changing the mouse cursor to another position could be prohibited in the text input field.

Another limitation is the extremely low number of participants. To get meaningful and generalizable results the study should be conducted with a lot more participants (n > 30) that are more diverse regarding their knowledge level of text input techniques and their keyboard usage.

Finally, the measurement of the input speed was not very good. Our definitions of when a word or sentence was finished works for simple sentences like the ones used in this study, but won't for more diverse and complex sentences. Better ways to track input would definitively be needed for this issue, for example the current word and even the characters of the word could be traversed one by one while the user enters the text, which could be used to check if the last character of a word or the last word of a sentence has been entered.

Results

Unfortunately, logging the data for the autocompletion conditions didn't work due to the implementation of the measuring logic and the way the autocompletion was implemented. In the `insertText()` method in the text_input_technique.py, where the replacing of the current word with the selected autocompletion option happens, the old input is deleted first which causes the measuring logic to fail as in the `eventFilter()` the keyboard input is checked key for key and the logic relies on the last entered char and the input history in general, which obviously isn't the same after replacing the current input. Because of this only the two conditions without autocompletion provided useful results so we couldn't compare our own input technique with them and can therefore neither reject nor accept our hypothesis.

[1] Qt Designer. (2021). *Qt Designer Manual*. Retrieved May 19, 2021, from https://doc.qt.io/qt-5/qtdesigner-manual.html

[2] PyQt. (2021). *What is PyQt?*. Retrieved May 19, 2021, from https://riverbankcomputing.com/software/pyqt

[3] QCompleter. (2021). *QCompleter Class*. Retrieved May 19, 2021, from https://doc.qt.io/qt-5/qcompleter.html

[4] TIGER Korpus. (2021). *TIGER Korpus*. Retrieved May 19, 2021, from https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger/

TIGER Korpus - Referenzen:

- Brants, Sabine, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic Interpretation of a German Corpus. Journal of Language and Computation, 2004 (2), 597-620.
- TIGER Project. 2003. TIGER Annotationsschema. Manuscript. Universität des Saarlands, Universität Stuttgart Universität Potsdam. July 2003.

The raw text used in the corpus is copyrighted by the Frankfurter Rundschau:

```
Druck- und Verlagshaus Frankfurt am Main GmbH
Verlag der Frankfurter Rundschau
Große Eschenheimer Straße 16-18
D-60313 Frankfurt am Main
```