WILDKEY: A PRIVACY-AWARE KEYBOARD TOOLKIT FOR DATA COLLECTION IN-THE-WILD

A WORKING PAPER

André Rodrigues
LASIGE, Faculdade de Ciências,
Universidade de Lisboa
afrodrigues@fc.ul.pt

André Santos
LASIGE, Faculdade de Ciências,
Universidade de Lisboa
afrodrigues@fc.ul.pt

Kyle Montague
NorthLab,
University of Northumbria
tjvg@di.fc.ul.pt

Hugo Nicolau
INESC-ID, Instituto Superior Técnico
Universidade de Lisboa
tjvg@di.fc.ul.pt

Tiago Guerreiro
LASIGE, Faculdade de Ciências,
Universidade de Lisboa
tjvg@di.fc.ul.pt

May 18, 2021

ABSTRACT

Touch data, and in particular text-entry data, has been mostly collected in the laboratory, under controlled conditions. While touch and text-entry data has consistently shown its potential for monitoring and detecting a variety of conditions and impairments, its deployment in-the-wild remains a challenge. In this paper, we present WildKey, an Android keyboard toolkit that allows for the usable deployment of in-the-wild user studies. WildKey is able to analyse text-entry behaviours through implicit and explicit text-entry data collection while ensuring user privacy. We detail each of the WildKey's components and features, all of the metrics collected, and discuss the steps taken to ensure user privacy and promote compliance.

Keywords Text-Entry · Touch Dynamics · In-the-wild · Smartphones · Data Collection

1 Introduction

The widespread use of smartphones has led to a massive increase in the generation of personal data, with text input being one of the most common tasks. People type to send messages, write emails, engage in social networks and much more. The data generated from these interactions has remarkable potential as a digital endpoint for disease detection and monitoring, to the quantified self movement, and biometrics. As a digital endpoint, text-entry metrics have been used to distinguish between people with Parkinson's Disease (PD) and control groups showing potential for early disease detection Arroyo-Gallego et al. [2017], Iakovakis et al. [2018a], Dhir et al. [2020], to assess stress Ciman et al. [2015], fatigue Al-Libawy et al. [2016], distinguish between patients with multiple sclerosis and controls Lam et al. [2020], and even for ubiquitous inebriation assessment Mariakakis et al. [2018]. In the field of user authentication, keystrokes dynamics have been used to discriminate among users detecting potential impostors, indicating that typing behaviour is highly personal Killourhy and Maxion [2009]. Preliminary work also suggests that differences between typing sessions can be associated with users' emotions Hadjidimitriou et al. [2017]. Despite all the potential shown by recent work, collecting this type of data, and particularly in-the-wild, raises several challenges in regards to required user effort, compliance, privacy, ease of deployment and study oversight.

There have been multiple approaches to collecting typing data in-the-wild. The most simple is explicitly prompting the user to do a specific task (e.g. transcription tasks) in a custom made application Reyal et al. [2015]. However this requires a lot of effort from the user, is not able to collect natural typing behaviour, nor provide spontaneous assessments, and overall one can expect less compliance as more data is requested from users. It has the benefit of mimicking controlled laboratory tasks which may provide data with less noise. A second approach is to collect everything the

user is typing by logging keypresses and touches Nicolau et al. [2017], which faces issues regarding user privacy and overall adherence to the study. A third solution is to only collect typing metrics that are not related with the content written such as flight time and hold times Iakovakis et al. [2018a]. The approach ensures user privacy at the cost of limiting the type of metrics one can extract from typing sessions. A similar approach is to obfuscate the text written, either by only parsing parts of the data Buschek et al. [2018] or by analysing the typing session and only storing an abstract representation of the sentences (e.g. “Noun Verb Adjective” or substituting every letter with an "M") Bemmann and Buschek [2020], Evans and Wobbrock [2012]. Depending on the method used, a different set of potential features can be extracted. All of the approaches described have advantages and disadvantages to them, and depending on the context, different ones, or a combination of several may be the ideal solution.

The WildKey toolkit was designed to support the collection of data both explicitly with prompted tasks, and implicitly providing the best of both. With the WildKey keyboard one can passively analyse all text written regardless of the application the user is in. Unlike prior approaches, WildKey neither stores raw textual data, nor it obfuscates typing sessions. Instead, for implicit data collecting we shifted the analyses that required access to the raw text to the device, and only calculated metrics with no potential to reconstruct the text are stored. The toolkit enables researchers and developers to tailor the WildKey data collecting to what best suits their study. The WildKey keyboard supports an unconstrained text-entry protocol MacKenzie et al. [2001] where users are free to write however they like including using suggestions, auto-correct and cursor changes Zhang and Wobbrock [2019]. It is able to provide all the traditional text-entry metrics on speed and error rates MacKenzie et al. [2001] and touch dynamic data that has been key in multiple studies Arroyo-Gallego et al. [2017], Iakovakis et al. [2018b], Lam et al. [2020] among others. As a toolkit for in-the-wild data collection it provides additional features that are commonly needed and even required to successfully run this type of studies. In addition to the text-entry tasks, one can create and schedule questionnaires and other custom made tasks such as the Alternate Finger Tapping assessment. Lastly, the toolkit has a Study Manager application that enables researchers/developers to easily schedule, deploy and oversee their active studies.

2 WildKey

The WildKey toolkit is composed of a *Keyboard Android app*, *a Study Management & Analytics app (React), and a NoSQL Database (Firebase)*. The toolkit was developed to enable researchers and developers to extend and deploy their own standalone ecosystems. The repository is open source, under the license Attribution 4.0 International (CC BY 4.0) and available at `https://techandpeople.github.io/keyboard/`.

The **WildKey keyboard** extends the Android Open Source Project Keyboard And [2021]. As such, it supports 26 languages, provides auto-correct, suggestions, customization options among many other traditional keyboard features. WildKey keyboard is responsible for collecting, and processing all the data inserted and storing its results on the defined cloud database storage. The details of the types of data collection and tasks available is discussed below in the *Data Collection*. A demo standalone version with local storage is available in the Playstore [toAppear].

The **Study Management & Analytics** application allows researchers to define and schedule their user studies and associate study tasks/schedules to registered users (details below in Study Management). Through it, we also offer a variety of features to support overseeing studies and quick visualizations of text-entry related metrics (*Metrics section*).

The database is responsible for storing all the information collected from each individual user and it is where all study configurations and schedules are stored. The keyboard app relies on these configurations to load and schedule tasks to its users, providing notifications, and other nudges (details below under Study Management).

The WildKey toolkit was designed to be **Privacy-Aware** in order to not only comply with the current standards for data protection, but also promote user compliance and adherence to study protocols. As such, during implicit text-entry collection (i.e. when the user is writing using the keyboard in any and all text fields regardless of application) no raw text is ever recorded on device storage or sent to the cloud, nor any data that would enable anyone to reconstruct the text-written. All text-entries that require content analysis are done locally on the device, and only processed data is sent to the cloud database. We highlight these and other design choices in the *Privacy-Aware* section.

2.1 Data Collection

The WildKey Keyboard app is prepared to collect three types of data: text-entry, questionnaires and custom made tasks. For text-entry, the keyboard is able to assess *Implicit Text-entry* data and to prompt users to do text-entry tasks collecting *Explicit Text-Entry* data. The keyboard app also enables developers to create *Questionnaires* to be answered within the app that accompanies the keyboard. Lastly, we designed the keyboard to be flexible and support the creation of *Custom-Made* tasks relevant for a specific study protocol.

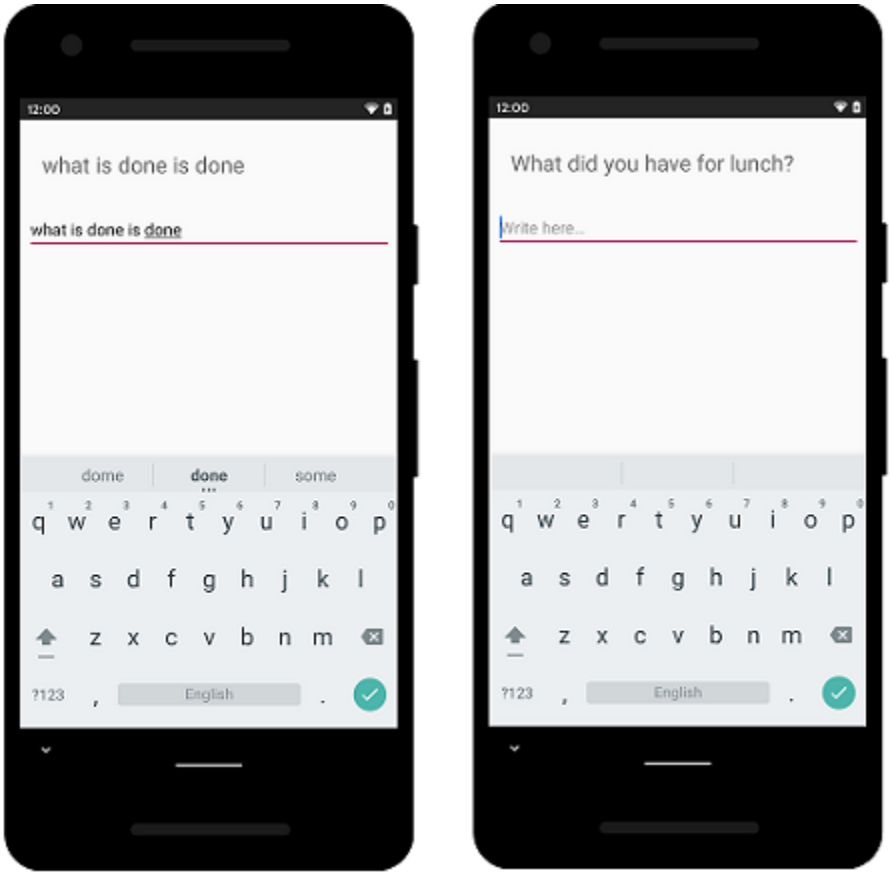

Figure 1: Transcription and Composition task example.
Two screens showing a transcription task and a composition task.

All data collected is sent via hypertext transfer protocol secure (HTTPS) and by default is prepared to store data in a NonSQL database (Firebase) with JavaScript Object Notation (JSON). The toolkit has an isolated data controller which enables developers to easily integrate an alternative data controller to match the database backend desired (e.g. MySQL).

When users first install the WildKey keyboard app they will be asked to create a user account by registering using an email address. WildKey will create a user profile and store in the database only a tokenID representative of the user with the email being used only for the first authentication. Additionally, Wildkey will collect information about the user device (i.e. operating system version, brand, model, screen dimensions and resolution).

2.1.1 Explicit Text-Entry Collection

There are two types of text-entry tasks supported by WildKey. The first is the standard transcription task used in traditional text-entry studies (i.e. asking the user to transcribe a set of phrases). The phrases to be provided to each user can be randomized and are provided when establishing the study protocol within the *Study Management*. Users are encouraged to type as accurately and quickly as possible. Transcription tasks allow us to know for certain what was the target intended phrase (since we provide it), thus enabling us to calculate error rate related metrics with high confidence. However, transcription tasks are not engaging for users, can be more demanding, do not portray natural typing behaviours and are not able to collect data spontaneously from users, nor for extended or frequent periods of time.

The second explicit text-entry task WildKey supports is Compositions. Users can be prompted at scheduled times to answer a sequence of questions following a similar protocol to the one described above. Unlike the Transcriptions tasks we do not have the ability to know for certain what was the intended sentence, or sentences the user was trying to write. Thus, to calculate metrics related with error rates we relied on the approach presented in Evans and Wobbrock

[2012] where target intent is calculated using a spell checker. Word by word, if it exists we consider it to be the equal to the user intent. In instances where the word does not exist we rely on the sentence written until the unknown written word (including it) and use the spell checker to predict the intended word, the top recommendation is chosen as the intended word to calculate all metrics. During the transcription tasks we also calculate and provide this calculated sentence based on the user input and spell checker to verify reliability. Composition tasks offer the possibility to ask questions relevant to the context of the user study providing additional key information (e.g. "What did you eat for lunch?"), are potentially more engaging for users, and typing behaviours will be more natural than ones triggered in transcriptions tasks. However, this type of task still faces similar challenges in its ability to collect data spontaneously, for extended or frequent periods of time.

WildKey supports an unconstrained text-entry protocol MacKenzie and Tanaka-Ishii [2007], where users are always free to correct any errors they encountered. Each text-entry trial (within an explicit task) has at the top of the screen the sentence/question to transcribe/answer, and an edit box below. The trial starts once the user writes the first letter, and ends when the user writes the last letter. The next trial is shown once the user confirms it on the keyboard. If it is the last one, users are sent to a completed task screen and shown when the next task is scheduled to.

Exclusively during explicit text-entry collection tasks, WildKey collects raw text, all user touch point data and calculates detailed error metrics (i.e. Omissions, Substitutions and Insertions - discussed below in Analytics).

2.1.2 Implicit Text-Entry Collection

When the keyboard is installed and active, all text written, regardless of where it is written, is analysed to calculate all the metrics described below in Analytics. The exceptions to the rule is when users are writing in password fields or inserting only numbers, where no metrics are calculated.

When the user opens the keyboard in any field, WildKey considers the first key entered to be the start of a text-entry trial, and the trial is ended once the keyboard is hidden again and the last key entered represents when the trial stopped. Furthermore, if more than a set amount of time elapses between two key inserts, we segment the sessions and consider it to be a new session. This allows us to segment text-entry sessions and calculate metrics for each of these trials (e.g. words per minute).

When the user opens the keyboard in a text edit field with text already present, and at some point during the trial it changes its cursor to the previously written text, we ignore the text-entry session and mark it as discarded. Since the text was written in a previous session we do not have the ability to compute any metrics (without compromising its reliability) and therefore the session is discarded. Nevertheless WildKey registers that the trial has been discarded and records some high level metrics for the session such as number of characters written.

Similar to composition tasks, during implicit data collection we do not know the user intended target sentence. As such we use the protocol described in the previous section to calculate user intent enabling us to assess error related metrics.

To preserve users' privacy and promote compliance, no text or any data that would enable its reconstruction is recorded during implicit collection. All metrics are calculated locally on the device and only metrics with no textual content are sent and stored in the cloud database.

2.1.3 Processing Text-Entry Data

While the user is writing, WildKey is creating an input stream buffer with all the text inputs. Simultaneously, Wildkey stores an array of the actions performed, an array of all cursor changes, and a list array of all the suggestions given by the spell checker with each letter entered. The array of actions is composed of corrections (i.e. deletes or substitutions) and entry actions Zhang and Wobbrock [2019]. The cursor changes array allows Wildkey to adjust the input stream at the end of the text-entry trial to account for non sequential changes to the text. In explicit text-entry collection with transcription tasks we rely on the target phrase provided to the users to compute all error related metrics. For implicit and composition tasks, the suggestions array enables WildKey to predict the user intent based on the spell checker predictions of the intended word. The input stream is then processed locally by Wildkey to compute all the metrics described below in Analytics; all other information is discarded. The calculated metrics are synchronized to the cloud database when an internet connection is available.

2.1.4 Questionnaires

A key feature to conduct in-the-wild studies is the ability to prompt users to answer and fill in short questionnaires and scales. As such, the Wildkey toolkit has the ability to schedule and prompt users to answer custom made questionnaires. The toolkit currently supports the creation of questionnaires with: slider scales, button scales, multiple choice list, single choice, time of day answers and open end questions (2). The questionnaires were developed to be easily

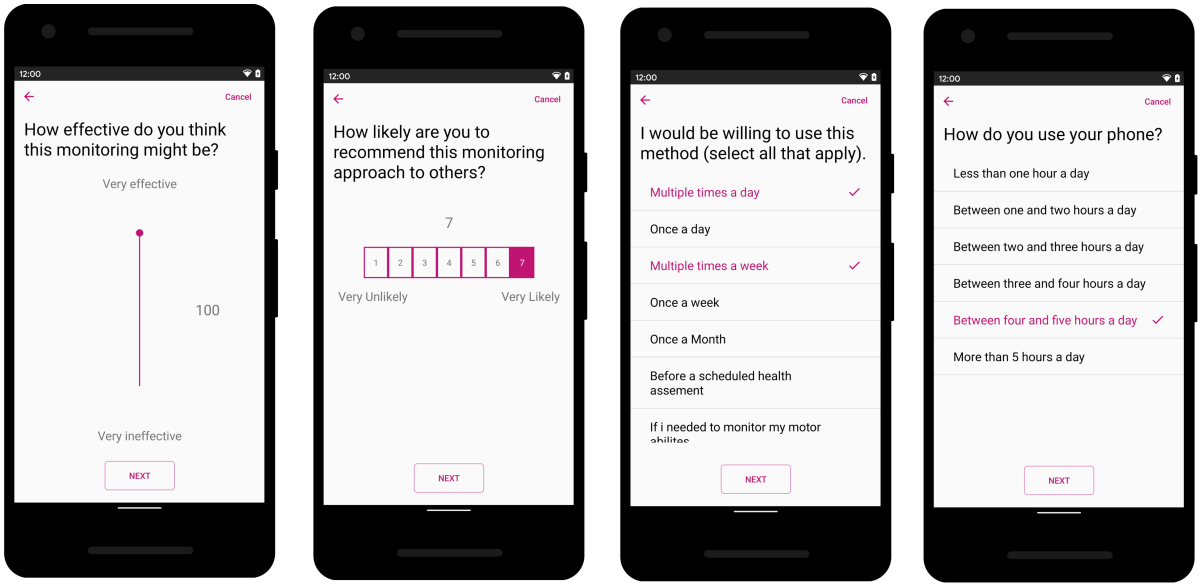

Figure 2: Four of the questions types available: slider scale, button scale, multiple choice and single choice. Four smartphone screens each showing one of the four of the questions types available: slider scale, button scale, multiple choice and single choice.

extendable with new types of questions/answers. Questions are provided one at a time in the sequence defined by the developer/researcher on the Study Management app. To enable users to control the flow of the questionnaire we relied on the library SurveyKit qui [2021] to enable back/cancel functionalities. Answers are collected and synchronized with the cloud database.

2.1.5 Custom Made Tasks

Similarly to questionnaires, Wildkey supports the creation of custom made tasks to schedule and deploy to its users. Currently, the toolkit has one example of such a task: the Alternate Finger Tapping test Lee et al. [2016].

2.2 Study Management

The React app has two key features: study management; and analytics. The application is currently deployable with local hosting and requires user authentication to access, create and edit studies. The study management is divided into User Studies, Tasks, and Users. In User Studies one can check and manage all the existing study schedules and create new study schedules based on the list of available tasks. In Tasks developers/researchers check existing tasks, and can create new tasks of the types described before (i.e. transcriptions, compositions, questionnaires or custom tasks). Lastly, in Users, each user is represented by a tokenID which can be associated with a study and with a schedule.

2.2.1 Creating Tasks

Through the study manager one can create four types of tasks. To create **1) transcription** or **2) compositions** tasks you can add the sequence of sentences/questions that you wish the user to do, you can add a timeout for the task (e.g. after 2 minutes the task does not ask for any additional sentences to be transcribed) and you can activate a randomizer of the order by which the sentences/questions are shown. To create **3) questionnaires**, first you have to create individual questions by selecting from the types available (i.e. slider scales, button scales, multiple choice list, single choice, time of day answers and open end questions) and filling in all required information (e.g. title, description, scales labels). Next, you can create a questionnaire by selecting all the questions that compose it; each question can be flagged as required. Lastly, for **4) Custom Made** tasks, the configurable parameters will depend on the task.

2.2.2 Creating a User Study

Through the React app one can create a user study that is composed of a title, start and end date, and one or more **schedules**. In turn, a schedule is composed of: a start and end date-time and a set of **timeframes**. A timeframe is an

interval of time within a day in which we want to make one or more tasks available. Each timeframe is composed of a start and end time of the interval, a title and description for the notification to be shown to users, and a task list which is composed of the tasks we wish to add for that particular interval (that we have created before, in tasks). An example is shown in Figure X where on the 4th of May between 8:00 and 20:00 the user could answer the questionnaire. When a schedule has been created, it can be edited, duplicated, deleted; it also has quick scheduling functions to allow shifting the scheduling by one day or week at a time to facilitate deploying multiple schedules where different users are onboarded on different days.

2.2.3 Study Dashboard

Through User Studies one can check all the schedules created and which users are attributed to which schedules. Each study has a **dashboard** that allows developers/researchers to quickly check the text-entry metrics collected by task type (e.g. average Words per Minute, Characters Written in transcriptions) associated with each user, and query for the metrics over specific period (e.g. Characters written in the last two days). The dashboard enables researchers/developers to quickly oversee the participants compliance and overall performance. As with the tasks, the Study Dashboard was developed to be easily expandable to accommodate other metrics which may be fundamental to the studies you desire to deploy.

2.2.4 User List

The user list is composed of token ids which represent the users which installed the keyboard application and have registered within the keyboard app. Each user can be associated with a study, and with a schedule within that study. Moreover, one can check a detailed view of the data that participants contributed and overall metrics (e.g. written characters). Similar to the study dashboard, one can also query for data over a specific period. Lastly, one can make a special request to download all data produced by a single user and delete all of it if requested.

2.3 User List

All metrics collected are calculated locally on the device before being sent out to the cloud database. By default, all data is stored in a JSON format and is readily available for download to post-process it wherever you desire. In table X you can find all the metrics collected and check in which of the types of data collection WildKey is able to calculate each of the metrics. We divided the collected and calculated metrics into five types: speed, errors, touch dynamics, action and character counts and other. The metrics shown in table X are only the ones associated with text-entry. All questionnaire answers produce data which is also stored in the same database. Below, we provide a brief description of all the text-entry metrics collected.

2.3.1 Speed

Speed metrics are related to how fast a user is writing. We are able to calculate speed in all types of tasks. Speed is calculated taking into account the time from the first entered character to the time of the last character entered. Whenever the keyboard is closed in implicit data collection, we consider the text-entry session to be completed and calculate the speed accordingly. In composition and transcription tasks, only when the user confirms the submission of the task, Wildkey computes speed and all other metrics.

- **Words per minute.** Calculated by (length of transcribed text)/(five characters per word) (60 seconds/trial time in seconds) Nicolau et al. [2017].
- **Time per word.** Calculated by (number of words)/(trial time in seconds)

2.3.2 Errors

Error rate metrics are an approximation, and a characterization of the errors users made while writing. In transcription tasks, we provide the user with the target phrase which enables us to calculate error rates without any caveats. In implicit collection and in composition tasks, WildKey relies on the spell checker to predict the user intent, as such one can always expect higher error rates due to abbreviations, use of multiple letters in a row to emphasize text (e.g. noooo) among others, all which will be potentially identified as errors (the process is previously detailed for compositions in the explicit text-entry collection section).

Wildkey calculates error rates for all types of tasks, but not all are available during implicit data collection due to some requiring significant computational power for longer writing sessions. Below we briefly detail all error rate metrics.

- **Corrected Error Rate.** Of the characters erased, the percentage that was erroneous. See Nicolau et al. [2017].

Data	Implicit	Transcription	Composition	Demo
Speed				
Words per Minute	✓	✓	✓	✓
Time per Word	✓	✓	✓	✓
Errors				
Corrected Error Rate	✓	✓	✓	✓
Uncorrected Error Rate	✓	✓	✓	✓
Total Error Rate	✓	✓	✓	✓
Insertion Error Rate		✓	✓	✓
Omission Error Rate		✓	✓	✓
Substitution Error Rate		✓	✓	✓
Error Correction Attempts	✓	✓	✓	✓
Touch Dynamics				
Flight Time	✓	✓	✓	✓
Hold Time	✓	✓	✓	✓
Touch Major/Minor	✓	✓	✓	✓
Touch Offset	✓	✓	✓	✓
Key Selected		✓	✓	
Motion Info		✓	✓	
Timestamp	✓	✓	✓	✓
Action and Character Counts				
Action Count	✓	✓	✓	✓
Correction Action Count	✓	✓	✓	✓
Entry Action Count	✓	✓	✓	✓
Number of Auto Corrects	✓	✓	✓	✓
Number of Changed Characters	✓	✓	✓	✓
Number of Selected Suggestions	✓	✓	✓	✓
Number of Written Characters	✓	✓	✓	✓
Number of Written Numbers	✓	✓	✓	✓
Number of Written Special Characters	✓	✓	✓	✓
Other				
Raw Text		✓	✓	✓
Intent Calculation	✓	✓	✓	✓
Intent Validation		✓		
Input Timestamp	✓	✓	✓	✓
Keyboard Language	✓	✓	✓	✓

Table 1: List of the Metrics captured by WildKey.

- **Uncorrected Error Rate.** Percentage of erroneous characters in the final transcribed sentence Nicolau et al. [2017], Wobbrock and Myers [2006].
- **Total Error Rate.** Of the characters entered, the percentage that was erroneous, corrected, or not Nicolau et al. [2017].
- **Insertion Error Rate.** Additional erroneous characters added (corrected and uncorrected). We can provide both individually by letter and aggregated measures. See Nicolau et al. [2017], Wobbrock and Myers [2006].
- **Omission Error Rate.** Corrected (characters that were missing at first but were backspaced and inserted) and Uncorrected Omitted characters in relation to the number of times the character was presented. We can provide both individually by letter and aggregated measures. See Nicolau et al. [2017], Wobbrock and Myers [2006].
- **Substitution Error Rate.** Ratio of substitutions to intentions. We can provide both individually by letter and aggregated measures. See Nicolau et al. [2017], Wobbrock and Myers [2006].
- **Error Correction Attempts.** Number of corrections sequences, meaning how many times the user started a sequence of correction actions (i.e. backspaces).

2.3.3 Touch Dynamics

WildKey collects a variety of touch related metrics (table X). For explicit tasks, all touch metrics are calculated since the text content the user writes is prompted by Wildkey and raw text content is collected. For implicit collection, no touch metrics are collected that would enable anyone to reconstruct the text written (i.e. keys selected, touch points).

- **Flight time**. Sequence of values corresponding to the time between the release of two key taps Arroyo-Gallego et al. [2017].
- **Hold Time**. Sequence of values of time spent touching the screen in each touch.
- **Touch Major/Minor**. Sequence of values of the TouchMajor (length of the major axis of an ellipse that represents the touch area) and the TouchMinor (length of the major axis of an ellipse that represents the touch area).
- **Touch offset**. Sequence of values of the differences in key centroids and hitpoint deviations (i.e., x and y offsets of touch gestures with regard to individual keys) Guerreiro et al. [2015], measured in pixels, and converted to cm given the device specifics.
- **Key selected**. The sequence of keypresses.
- **Motion info**. Sequence of all the touch motions detected by Android and their timestamp (i.e. down, move, up).

2.3.4 Action and Character Counts

To enable us to better characterize the users text-entry behaviours, Wildkey collects a wide variety of action and character counts.

- **Action Count**. Total number of actions performed (i.e. corrections and entry actions).
- **Correction Action Count**. Total number of individual actions that corrected input (i.e. substitutions and deletions).
- **Entry Action Count**. Total number of individual actions that produced an input. Using a suggestion counts as a single entry action.
- **Number of Auto Corrects**. Total number of auto-corrects.
- **Number of Changed Characters**. Total number of changes to the input stream.
- **Number of Selected Suggestions**. Number of times suggestions were selected.
- **Number of Written Characters**. Total number of written characters.
- **Number of Written Numbers**. Total number of written numbers.
- **Number of Written Special Characters**. Total number of written special characters.

2.3.5 Other

In other we aggregate the data/metrics that are related with intent, raw text and other device/system characteristics.

- **Raw Text.** In tasks where we explicitly prompt the user we collect the input stream of all user interactions with the keyboard.
- **Intent Calculation.** When we do not provide the target sentence for the user to transcribe, Wildkey estimates the user intent. The final target phrase is calculated and available for every text-entry session.
- **Intent Validation.** In transcription tasks, we also calculate the user's final target phrase through intent calculation. Although we rely on the target phrase we provide to generate all metrics, the calculation of the intent allows us to have a baseline of the intent calculation performance which we refer to as intent validation.
- **Input Timestamp.** A sequence of all touchdown input timestamps, including selecting suggestions and making cursor changes.
- **Keyboard Language.** For every session we record the current language of the keyboard and any changes during the text-entry session.

2.4 Privacy-Aware

Smartphones are becoming an extension of oneself Park and Kaye [2019] and data privacy has become of the utmost importance. Text content can be one of the most private types of data users generate and approaches that seek to collect this type of information can expect to be met with resistance by some users and overall suffer from lower levels of adherence and compliance. Aware of the challenges WildKey was devised to not store any textual content outside the specific tasks it asks users to perform. Moreover, all metrics that would enable the reconstruction of the text content are not available when analysing users text input behaviours implicitly. WildKey calculates all metrics locally on the device and only stores the results of those calculations in the cloud. Furthermore, when the user is inserting only numbers or in passwords fields no analysis is conducted. To ensure users are in control of the data they are sharing, WildKey has an always available button on the top left of the keyboard to activate an "incognito mode", which resets every time the keyboard is closed. When active no data is analysed. Lastly, as with other Android keyboards, users can quickly swap to another active keyboard by holding the space bar for a quick options menu.

WildKey currently relies on Firebase services for its cloud storage which encrypts data in transit using HTTPs and logically isolates customer data. All users data is pseudo-anonymised within the database where users are only identifiable by their tokenID. Cross reference between tokenIDs and users identification is not stored in the same database.

To facilitate GDPR compliance in regards to the 'right to be forgotten' and the right users have to request their data, in the Study Management application under users, WildKey enables researchers to quickly download all data from a specific user, and delete it all if necessary.

Lastly, in the github repository for the WildKey React app we provide a mock up report (pdf file) which shows the type of data WildKey is able to collect in a comprehensive manner. The report can be leveraged by researcher/developers to enable users to make a conscious choice when deciding to use the keyboard or participate in a user study.

2.5 Limitations

WildKey is currently only available on Android, supporting devices with operating systems 6.0 and above which covers about 84.9% of all devices according to Android Platform - API Version Distribution. The number of supported devices will continue to grow as deprecated devices cease to function and get replaced.

For composition and implicit text collection we rely on intent prediction to calculate error related metrics which artificially increases the error rates due to a variety of text-entry behaviours people have in informal conversations (e.g. abbreviations). Moreover, the intent prediction is only as accurate as the used dictionary and spell checker. For words that exist in the dictionary, all entered are considered correct, but for words that are not detected we use the spell checker best prediction, which may vary across different spell checkers and dictionaries. We relied on the open source Android OS dictionary and spell checker which provides support for 26 languages. The quality of the intent prediction can additionally vary among different languages. While WildKey currently does not provide a streamline process to change or add dictionaries and/or spell checkers it is possible for developers to customize these options.

Although the toolkit was designed to support a wide variety of in-the-wild study protocols it may require additional development efforts for custom-tailored deployments. The demo available on the Playstore provides a standalone version with local storage which facilitates deployment, but complicates data retrieval and study oversight. To leverage the full potential of the toolkit developers/researchers have to deploy the WildKey toolkit ecosystem which currently requires some degree of technical expertise.

3 Conclusions

The data one can collect from typing sessions has continually shown its potential for a variety of different domains. As a complex task that combines motor and cognitive functions, the features one can extract can be explored for disease detection and monitoring, for biometrics, personalized assistive technology, and within the quantified self movement. To support and promote further explorations we developed and made available to all the WildKey Toolkit with its first release. The toolkit is currently being leveraged in multiple studies with topics from privacy and compliance, to fatigue, to monitoring motor fluctuations in neurodegenerative diseases. The toolkit is an ongoing open source project and we welcome contributions. New features will continue to be added as new requisites appear to support studies with different goals and in different contexts.

We are very keen to hear about your work. Please reach out if you decide to use the toolkit, have any questions or desires for additional features

4 Conclusions

We would like to thank our collaborators from OpenLab Newcastle as the first external team to use the keyboard toolkit providing us with valuable feedback and contributions to the project. This project was partially supported by FCT through LASIGE Research Unit funding, ref. UIDB/00408/2020 and ref. UIDP/00408/2020, and by the IDEA-FAST project which has received funding from the Innovative Medicines Initiative 2 Joint Undertaking under grant agreement No. 853981. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and EFPIA and associated partner.

References

Teresa Arroyo-Gallego, María Jesus Ledesma-Carbayo, Álvaro Sánchez-Ferro, Ian Butterworth, Carlos S. Mendoza, Michele Matarazzo, Paloma Montero, Roberto López-Blanco, Verónica Puertas-Martín, Rocío Trincado, and Luca Giancardo. Detection of motor impairment in parkinson's disease via mobile touchscreen typing. *IEEE Transactions on Biomedical Engineering*, 64(9):1994–2002, 2017. doi:10.1109/TBME.2017.2664802.

Dimitrios Iakovakis, Stelios Hadjidimitriou, Vasileios Charisis, Sevasti Bostantjopoulou, Zoe Katsarou, Lisa Klingelhoefer, Heinz Reichmann, Sofia B. Dias, José A. Diniz, Dhaval Trivedi, K. Ray Chaudhuri, and Leontios J. Hadjileontiadis. Motor impairment estimates via touchscreen typing dynamics toward parkinson's disease detection from data harvested in-the-wild. *Frontiers in ICT*, 5:28, 2018a. ISSN 2297-198X. doi:10.3389/fict.2018.00028. URL `https://www.frontiersin.org/article/10.3389/fict.2018.00028`.

Neil Dhir, Mathias Edman, Álvaro Sanchez Ferro, Tom Stafford, and Colin Bannard. Identifying robust markers of Parkinson's disease in typing behaviour using a CNN-LSTM network. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 578–595, Online, November 2020. Association for Computational Linguistics. doi:10.18653/v1/2020.conll-1.47. URL `https://www.aclweb.org/anthology/2020.conll-1.47`.

Matteo Ciman, Katarzyna Wac, and Ombretta Gaggi. isensestress: Assessing stress through human-smartphone interaction analysis. In *2015 9th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, pages 84–91, 2015. doi:10.4108/icst.pervasivehealth.2015.259280.

Hilal Al-Libawy, Ali Al-Ataby, Waleed Al-Nuaimy, Majid A. Al-Taee, and Qussay Al-Jubouri. Fatigue detection method based on smartphone text entry performance metrics. In *2016 9th International Conference on Developments in eSystems Engineering (DeSE)*, pages 40–44, 2016. doi:10.1109/DeSE.2016.9.

KH Lam, KA Meijer, FC Loonstra, EME Coerver, J Twose, E Redeman, B Moraal, F Barkhof, V de Groot, BMJ Uitdehaag, and J Killestein. Real-world keystroke dynamics are a potentially valid biomarker for clinical disability in multiple sclerosis. *Multiple Sclerosis Journal*, 0(0):1352458520968797, 2020. doi:10.1177/1352458520968797. URL `https://doi.org/10.1177/1352458520968797`. PMID: 33150823.

Alex Mariakakis, Sayna Parsi, Shwetak N. Patel, and Jacob O. Wobbrock. Drunk user interfaces: Determining blood alcohol level through everyday smartphone tasks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 234:1–234:13, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5620-6. doi:10.1145/3173574.3173808. URL `http://doi.acm.org/10.1145/3173574.3173808`.

Kevin S. Killourhy and Roy A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, pages 125–134, 2009. doi:10.1109/DSN.2009.5270346.

Stelios Hadjidimitriou, Dimitrios Iakovakis, Vasileios Charisis, Sofia B. Dias, José A. Diniz, Julien Mercier, and Leontios J. Hadjileontiadis. On capturing older adults' smartphone keyboard interaction as a means for behavioral change under emotional stimuli within i-prognosis framework. In Margherita Antona and Constantine Stephanidis, editors, *Universal Access in Human–Computer Interaction. Design and Development Approaches and Methods*, pages 346–356, Cham, 2017. Springer International Publishing. ISBN 978-3-319-58706-6.

Shyam Reyal, Shumin Zhai, and Per Ola Kristensson. Performance and user experience of touchscreen and gesture keyboards in a lab setting and in the wild. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 679–688, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450331456. doi:10.1145/2702123.2702597. URL `https://doi.org/10.1145/2702123.2702597`.

Hugo Nicolau, Kyle Montague, Tiago Guerreiro, André Rodrigues, and Vicki L. Hanson. Investigating laboratory and everyday typing performance of blind users. *ACM Trans. Access. Comput.*, 10(1), March 2017. ISSN 1936-7228. doi:10.1145/3046785. URL `https://doi.org/10.1145/3046785`.

Daniel Buschek, Benjamin Bisinger, and Florian Alt. Researchime: A mobile keyboard application for studying free typing behaviour in the wild. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*,

CHI '18, page 1–14, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi:10.1145/3173574.3173829. URL https://doi.org/10.1145/3173574.3173829.

Florian Bemmann and Daniel Buschek. Languagelogger: A mobile keyboard application for studying language use in everyday text communication in the wild. *Proc. ACM Hum.-Comput. Interact.*, 4(EICS), June 2020. doi:10.1145/3397872. URL https://doi.org/10.1145/3397872.

Abigail Evans and Jacob Wobbrock. Taming wild behavior: The input observer for obtaining text entry and mouse pointing measures from everyday computer use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 1947–1956, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450310154. doi:10.1145/2207676.2208338. URL https://doi.org/10.1145/2207676.2208338.

I. Scott MacKenzie, Tatu Kauppinen, and Miika Silfverberg. Accuracy measures for evaluating computer pointing devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, page 9–16, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133278. doi:10.1145/365024.365028. URL https://doi.org/10.1145/365024.365028.

Mingrui Ray Zhang and Jacob O. Wobbrock. Beyond the input stream: Making text entry evaluations more flexible with transcription sequences. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 831–842, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368162. doi:10.1145/3332165.3347922. URL https://doi.org/10.1145/3332165.3347922.

Dimitrios Iakovakis, Stelios Hadjidimitriou, Vasileios Charisis, Sevasti Bostantzopoulou, Zoe Katsarou, and Leontios J. Hadjileontiadis. Touchscreen typing-pattern analysis for detecting fine motor skills decline in early-stage parkinson's disease. *Scientific Reports*, 8(1), May 2018b. doi:10.1038/s41598-018-25999-0. URL https://doi.org/10.1038/s41598-018-25999-0.

Android open source project. https://source.android.com/, 2021. (Accessed on 05/18/2021).

I. Scott MacKenzie and Kumiko Tanaka-Ishii. *Text entry systems mobility, accessibility, universality*. Boston, 2007.

quickbirdstudios/surveykit: Android library to create beautiful surveys (aligned with researchkit on ios). https://github.com/quickbirdstudios/SurveyKit, 2021. (Accessed on 05/18/2021).

Chae Young Lee, Seong Jun Kang, Sang-Kyoon Hong, Hyeo-Il Ma, Unjoo Lee, and Yun Joong Kim. A validation study of a smartphone-based finger tapping application for quantitative assessment of bradykinesia in parkinson's disease. *PLOS ONE*, 11(7):1–11, 07 2016. doi:10.1371/journal.pone.0158852. URL https://doi.org/10.1371/journal.pone.0158852.

Jacob O. Wobbrock and Brad A. Myers. Analyzing the input stream for character- level errors in unconstrained text entry evaluations. *ACM Trans. Comput.-Hum. Interact.*, 13(4):458–489, December 2006. ISSN 1073-0516. doi:10.1145/1188816.1188819. URL https://doi.org/10.1145/1188816.1188819.

João Guerreiro, André Rodrigues, Kyle Montague, Tiago Guerreiro, Hugo Nicolau, and Daniel Gonçalves. Tablets get physical: Non-visual text entry on tablet devices. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 39–42, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450331456. doi:10.1145/2702123.2702373. URL https://doi.org/10.1145/2702123.2702373.

Chang Sup Park and Barbara K. Kaye. Smartphone and self-extension: Functionally, anthropomorphically, and ontologically extending self via the smartphone. *Mobile Media & Communication*, 7(2):215–231, 2019. doi:10.1177/2050157918808327. URL https://doi.org/10.1177/2050157918808327.