Measuring Mobile Text Entry Performance and Behaviour in the Wild with a Serious Game

Christoph Wimmer, Richard Schlögl, Karin Kappel, Thomas Grechenig
TU Wien, Research Group for Industrial Software (INSO)
Vienna, Austria
christoph.wimmer|richard.schloegl|karin.kappel|thomas.grechenig@inso.tuwien.ac.at

ABSTRACT

Entering text is a fundamental part of how we interact with computing devices. The predominant form of text input on smartphones are virtual keyboards, which provide greater flexibility and customization options compared to hardware keyboards. Mobile text entry performance has been widely studied in HCI research, with most experiments being conducted in a controlled, artifical lab environment. To escape the boundaries of the lab and observe mobile text entry behaviour in a realistic and natural environment, we developed and publicly released the game Hyper Typer on Google Play Store. Hyper Typer is a serious research game for measuring mobile text entry performance and behaviour on a large scale in the real world. Publishing the game on Google Play Store resulted in a total of 2,359 valid transcribed phrases with 71,963 keystrokes. In this paper we discuss the game design of Hyper Typer, present the results collected over a time period of one year after the release of the game and reflect on the advantages, disadvantages and challenges of deploying a research game publicly.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in HCI**; *Empirical studies in ubiquitous and mobile computing*; *Ubiquitous and mobile computing design and evaluation methods.*

KEYWORDS

Serious games; games with a purpose; text entry; text input; evaluation methodology; crowd science.

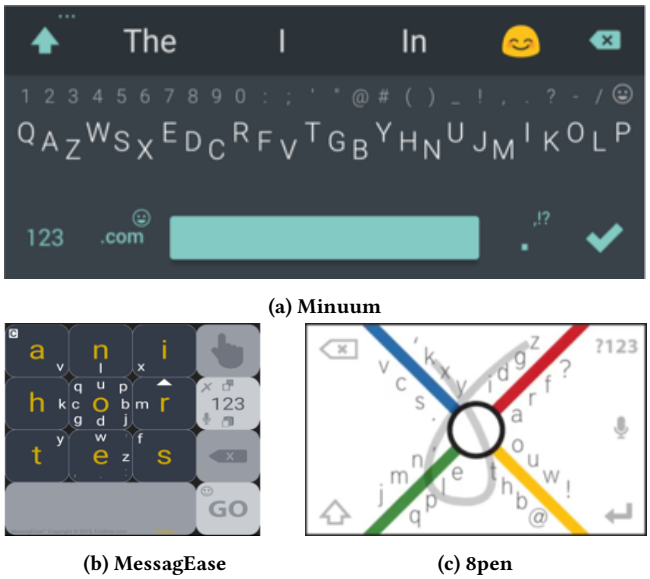
ACM Reference Format:
Christoph Wimmer, Richard Schlögl, Karin Kappel, Thomas Grechenig. 2019. Measuring Mobile Text Entry Performance and Behaviour in the Wild with a Serious Game. In *MUM 2019: 18th International Conference on Mobile and Ubiquitous Multimedia (MUM 2019), November 26–29, 2019, Pisa, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3365610.3365633

1 INTRODUCTION

Entering text is one of the most fundamental ways of how we interact with our mobile devices. Many apps rely on text input in order to fulfill their purpose, be it to send a message, fill out an input field in a form, edit documents or jot down a quick note. In contrast to earlier generations of mobile devices which typically featured physical buttons for text entry, the advent of touchscreens on modern smartphones allows for more flexibility and with it a proliferation of different text entry methods as the means of text entry are no longer bound to the physical shape and form of the device.

Modern smartphone operating systems generally rely on virtual software keyboards for text input on touchscreens. These virtual keyboards can be easily customized or changed. Aside from Gboard, the default Android keyboard provided by Google, many Android device manufacturers such as Samsung, LG and Huawei ship their own proprietary keyboard implementations with their Android smartphones. Furthermore, as virtual keyboards are not bound to one specific shape and form, many keyboard implementations provide fine-grained customization options to their users regarding both appearance and behaviour.

In addition to this multitude of virtual keyboard designs provided by smartphone manufacturers out of the box, the two dominant smartphone operating systems in use today, Android and iOS, both allow their users to swap text entry mechanisms according to their preferences by downloading alternative virtual keyboards and other text entry mechanisms from their respective app stores to replace



(a) Minuum

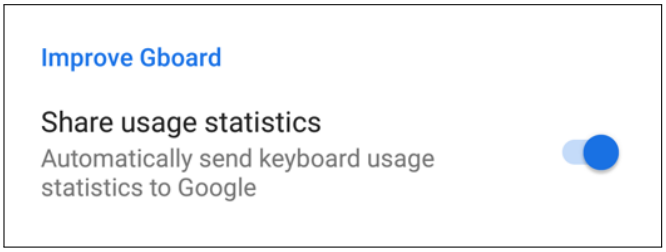
(b) MessagEase

(c) 8pen

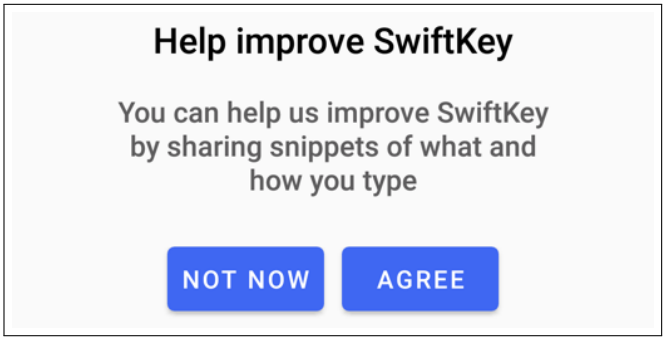
Figure 1: Examples of innovative and unconventional keyboard designs

MUM 2019, November 26–29, 2019, Pisa, Italy

ACM ISBN 978-1-4503-7624-2/19/11…$15.00
https://doi.org/10.1145/3365610.3365633



(a) Gboard usage data sharing setting



(b) SwiftKey usage data sharing opt-in

Figure 2: Examples of keyboard usage data sharing

the default implementations. Even though the default virtual keyboards included on today's smartphones are rather similar designs based on the QWERTY layout and variations thereof, the flexibility afforded by the proliferation of virtual keyboards has proven a fertile ground for unconventional and innovative text entry mechanisms such as Minuum [42] (see Figure 1a) or MessagEase [5] (see Figure 1b). Furthermore there are text entry mechanisms which eschew the predominant keyboard metaphor entirely such as handwriting recognition and gestural input, e.g. 8pen [39] (see Figure 1c).

However, this abundance of choice available on modern smartphones raises the question of how to assess the utility and usability of different text entry mechanisms and whether they provide a tangible benefit to users. Detailed usage metrics and performance measures may be readily available to platform providers and keyboard developers, as many of these text entry mechanisms collect usage data and statistics on behalf of their developers (e.g. Google Gboard or Microsoft SwiftKey, see Figures 2a and 2b). Some of them even expose certain usage data to their users such as Microsoft SwiftKey. However, this usage data on text entry behaviour and performance can be hard to come by for independent researchers.

For this reason and in order to gain a better understanding of real-world text entry performance in a natural setting we developed a serious game named Hyper Typer [32]. Hyper Typer is a mobile typing game for Android devices with the purpose of collecting text entry performance data from players of the game on an ongoing basis and without direct supervision for exploratory data analysis. In contrast to a controlled lab experiment, players of Hyper Typer can play the game voluntarily whenever they feel motivated, in their natural environment and on their personal mobile device with all their text entry and keyboard settings intact as they are used to. In this paper we expand our previous work introducing Hyper Typer [33] with findings from more extensive evaluation.

2 RELATED WORK

Empirical studies and experiments in the field of HCI research typically require volunteers and it can be challenging for scientists and researchers to recruit a sufficient number of participants. To address this challenge, some researchers have turned to crowdsourcing for recruiting participants on platforms such as Amazon Mechanical Turk [6, 17]. Serious games for research are a promising crowdsourcing approach for turning players into voluntary participants in experiments and field studies by incentivizing their participation with entertaining and engaging gameplay. For this approach to be successful, it is necessary that the tasks presented to participants can be embedded in fun and engaging gameplay mechanics.

2.1 Serious Games

Serious games are games designed for a primary purpose other than pure entertainment [27]. As such, serious games feature both a game dimension and a serious dimension, whereas pure entertainment games consist only of a game dimension. Serious games have been developed for a wide and diverse range of purposes across different domains such as healthcare games, educational games, corporate games, political games, advertising games as well as research games [4]. While the entertainment purpose of a serious game is to the immediate benefit of the player, the desired serious purpose can be either beneficial to the players of the game (e.g. by improving learning success in case of an educational game or for rehabilitation purposes in case of a healthcare game), to the developers and publishers of a game (e.g. by communicating a message in case of a political or advertising game) or both.

A concept related to serious games are games with a purpose (GWAPs). While the name itself implies that GWAPs closely align with the definition of "games that do not have entertainment, enjoyment or fun as their primary purpose" [27], the term was initially introduced with a narrower and more specific definition of meaning, originally referring to games "in which people, as a side effect of playing, perform tasks computers are unable to perform" [41]. By this definition, GWAPs should be considered a specific subcategory within the broad field of serious games. Hyper Typer as a research game for data collection can be considered both a serious game as well as a game with a purpose, as its purpose is to collect empirical data which cannot be obtained without human participation.

Typing games are a niche in the overall video game market, both on desktop PC as well as on mobile devices such as smartphones. Typing games typically aim to both entertain their players and help them improve their typing skills in a fun and motivating way. As such, many typing games qualify as serious games as they often serve an educational or training purpose. Typing games have been used for scientific and research purposes. Kristensson and Zhai [19] developed a game to teach users shape writing, a text entry technique where users draw shapes which connect all the letters in the desired word across a graphical representation of a keyboard. Rudchenko et al. [31] developed Text Text Revolution, a typing game to provide targeting practice to users and generate training data for

key-target resizing as a side effect of playing the game. Through a user study and simulation experiment they demonstrated that participants improved in accuracy over time and that the training data for key-target resizing could be used to reduce error rate by 21.4 %. Henze et al. [13] employed a typing game to collect more than 47 million keystroke events from 72,945 installations. They used this data to identify a systematic skew and derived a function to compensate for it by shifting touch events. Vertanen et al. [40] developed Text Blaster, a multiplayer shoot'em up game where players type sentences on a mobile device's touchscreen keyboard, intended to investigate performance and design aspects of touchscreen text entry mechanisms. They found that the competitive nature of gameplay encouraged players to enter text both quickly and accurately and hypothesized "that these properties might make Text Blaster an ideal platform for conducting both laboratory and crowdsourced text entry experiments".

2.2 Large-scale HCI Experiments

Beyond the typing game genre, prior studies have employed serious research games and apps released publicly on app stores to conduct large-scale HCI experiments or collect data in the field. Henze et al [11] used a game to evaluate three visualization techniques for off-screen objects from 3,934 installations. Henze et al. [12] later used the touch-targeting game Hit It! to collect more than 120 million touch events from 91,731 installations to determine error rates for different target sizes and screen locations, finding that touch positions are systematically skewed and deriving a compensation function based on this data. The game Hit It! was also used in a number of additional studies and experiments [8].

Releasing and disseminating an app for research purposes through open marketplaces such as the Apple App Store or Google Play Store is fraught with its own challenges and a number of case studies report on the varying successes and failures experienced by researchers in this endeavor. McMillan et al. [24] reflected on their experiences in running a large-scale trial of a mobile game with more than 40,000 users on the Apple App Store and discussed benefits and potential shortcomings. They warned researchers interested in their approach to "[e]xpect low percentages of uptake and participation" and offered guidance on how to manage a large and disparate user base. McMillan et al. [25] later compared different software distribution methods via the official Apple App Store and third party software repositories. Henze et al. [10] reflected on five experiments conducted by publishing apps on the Android Market and identified factors that account for the success of experiments using mobile app stores. They concluded that "experiments in the market allow to gain insight that otherwise would only be possible to obtain with an enormous amount of effort" but caution that such efforts are not automatically successful.

The use of research apps publicly released on app stores raises questions regarding the validity of results, both internal and external [8–10]. Just as HCI research experiments in general, studies and experiments concerning text entry performance evaluation are commonly conducted in a controlled environment such as a lab setting. This approach increases internal validity, as researchers are better able to control confounding factors such as environmental conditions, leading to more reliable results. However, the downside to this approach is that it can limit external validity, meaning the generalizability of results to real contexts beyond the artificial, controlled environment of the experiment. Henze et al. caution that "[e]xperiments [...] can have a high internal validity and be very reliable without necessarily having much to do with humans' real life behaviour" [8] and have argued that "applications evaluated in mobile application stores reach a large number of users that use the App in "natural" settings" [10] and "that app stores are powerful tools for increasing the ability to generalize results when studying mobile and pervasive systems" [9].

Large scale user studies using publicly released apps raise questions regarding ethical issues [26, 28] as well. In line with recommendations found in literature [28] and to meet our ethical obligations towards participants, we limit data collection to anonymous data. Participants are notified of their participation in a research project and what data is collected when first starting the app in order to gain their informed consent. Data collection is limited to as much as necessary and as little as possible to reach our research goals.

3 GAME DESIGN AND IMPLEMENTATION

The game design of Hyper Typer was guided and informed by game design theory such as the MDA framework [14] and the Human Computing to Video Games (HCtVG) model [3], existing typing games such as TypeRacer [38] and ZType [37] and established research best practices for text entry evaluation in general [18, 23, 29, 35] and specifically on mobile devices [1, 2, 16, 21, 30].

Hyper Typer aims to benefit both the players of the game as well as researchers: Aside from its entertainment purpose, Hyper Typer allows players to improve their typing skills through practice and to assess their text entry performance in a quantifiable way by means of a scoring mechanism that takes both text entry speed as well as accuracy into account. In addition, the detailed measurement of text entry performance allows researchers to gain a better understanding of real world text entry performance from a large number of players across a variety of different devices and text entry mechanisms.

3.1 Design Process and Design Considerations

The design and development of a serious game can be challenging as it requires a broad and diverse set of skills. As mentioned, serious games comprise of both a game dimension and a serious dimension, and designing a serious game for research purposes thus not only requires proficiency in research methodology, but in game design as well. When designing serious games for research purposes it is necessary to align the research objectives with the game design and that the tasks to be completed by the player can be embedded in fun and engaging gameplay mechanics.

The HCtVG model for integrating human computing into video games [3] suggests a process that begins with the deconstruction of research objectives into desired outputs before integrating the research activity into the game loops. The desired outputs for Hyper Typer are performance measures of text entry performance such as words per minute and error rate, which are well established in the literature and related work on text entry experiments. The design of Hyper Typer and its core gameplay loop was primarily informed

by the design of existing text entry experiments and associated recommendations [1, 2, 18, 23, 29, 35].

The primary purpose of Hyper Typer is the ongoing collection of real-world data and performance measures of typing behaviour without direct supervision of study participants. To achieve this goal, the game was designed to be self-explanatory and suitable for unsupervised use and public dissemination on app stores in order to reach as many players as possible. Nevertheless, the game can also be used for conducting lab experiments in a controlled setting.

Text entry experiments generally task their participants with transcribing a number of predefined phrases chosen from a representative phrase set "as quickly and as accurately as possible" [18]. As such, the core task embedded in the game design is the transcription of predefined phrases. This transcription task is similar to the core gameplay loop of many existing typing games and as such has proven itself to be suitable for this kind of game. All subsequent design decisions were informed by this core transcription task and where trade-offs between experiment design and game design were necessary, decisions were favored that maintained the integrity of the research objective to ensure the validity of the collected data.

Text entry experiments typically require participants to enter more than a single phrase in order to gather a sufficient amount of data for subsequent analysis (e.g. 10 phrases per condition across three conditions for a total of 30 phrases per participant in [1] and [2]). As such, the structure of the game should also incentivize players to transcribe multiple phrases. On the other hand, mobile smartphone games generally favor a structure which enables short bursts of gameplay to allow for spontaneous, intermediate play in a variety of everyday situations without placing undue burden on the player. To balance this trade-off, Hyper Typer was designed to require players to transcribe only a relatively small number of phrases consecutively in order to finish a game round. This allows players to finish a single game round within a few minutes. However, to encourage players to return to the game for repeated play and thus provide more substantial amounts of data, the game employs mechanisms such as highscore lists and achievements with the intention of providing additional goals and thus motivation for players to keep playing.

To support realistic text entry behaviour, Wobbrock [23] describes three requirements for the unconstrained text entry evaluation paradigm [35, 43]:

- All printable characters are accepted as legitimate input during transcription. This requirement specifically allows for entering incorrect characters.
- Using backspace is the only means of correcting errors.
- No error beeps or other intrusions affect text entry.

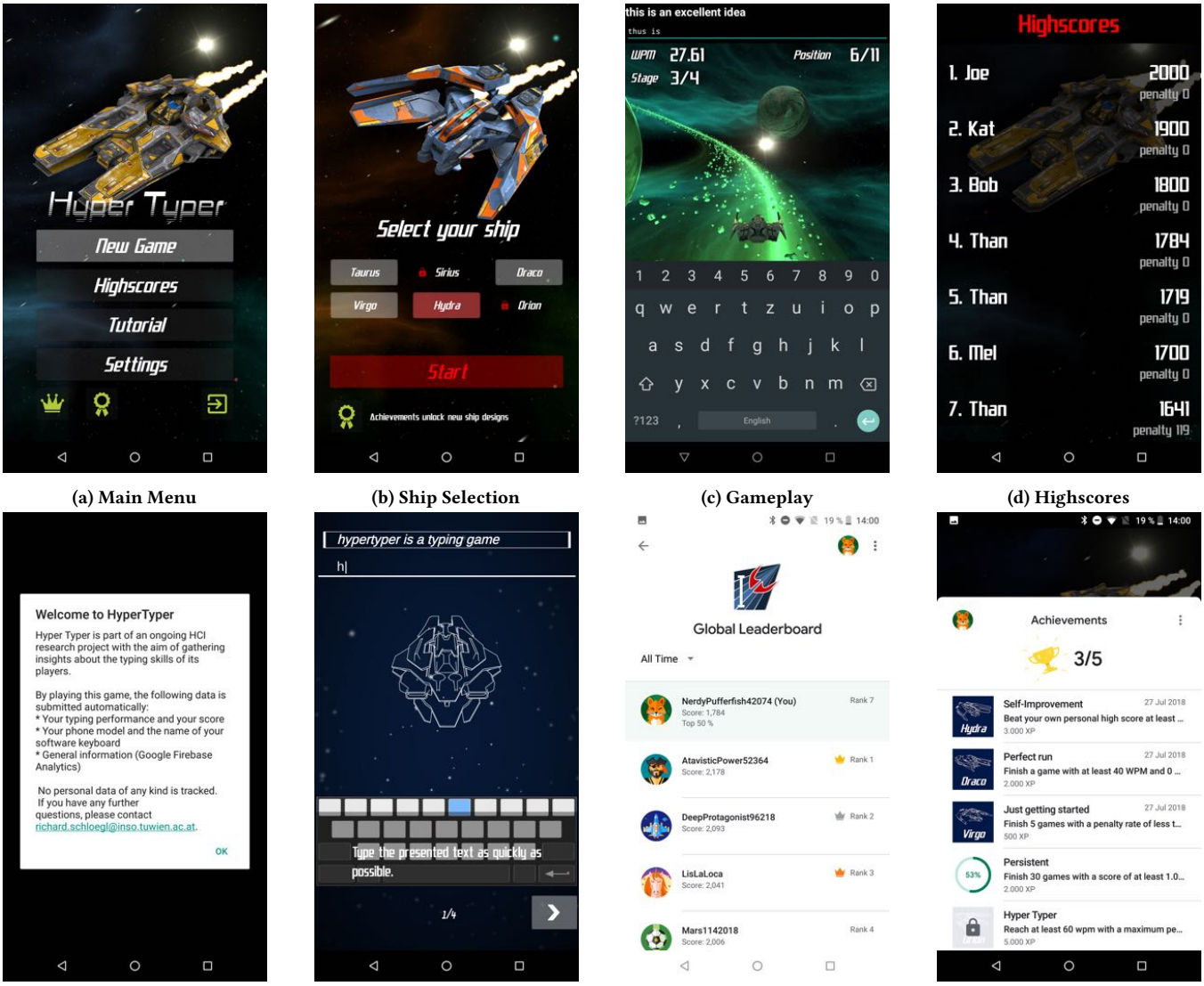
Some existing typing games violate these requirements by disallowing incorrect input [38], ignoring incorrect characters [37] or otherwise limiting error correction mechanisms. Rudchenko et al. [31] observed that typing games which force players to enter the correct input in order to progress tend to influence the player's behaviour to slow down and type more carefully. To ensure realistic and natural typing behaviour in Hyper Typer, the game adopts the unconstrained text entry evaluation paradigm. Players are allowed to enter incorrect characters and they can correct these errors using the backspace key. No intrusive feedback regarding the correctness of input is provided during the transcription task and feedback is delayed until the player has finished the transcription of a phrase.

Hyper Typer presents players with their familiar Android device keyboard as configured in the Android system settings and keeps all their preferences and configuration options intact. This is in contrast to some other studies [13] and typing games (e.g. ZType [37]), which provide their own custom keyboard implementation for text entry. Henze et al. [13] specifically mention that in order to "increase the study's internal validity, the same keyboard is used for all devices" and they created their own keyboard implementation based on the standard Android keyboard for this purpose. The development of a custom keyboard grants game designers, developers as well as researchers more fine-grained control by providing a standardized, uniform method of text entry for players. However, providing a custom keyboard implementation has a high risk that players might be unfamiliar with the presented keyboard, forcing them to adapt to the new keyboard and thus negatively impacting their text entry performance. In addition, custom keyboards are sometimes limited to specific layouts such as the QWERTY layout, which may be challenging for players unfamiliar with the presented layout. As such, providing a custom keyboard implementation is contrary to our stated purpose of collecting data about realistic and natural typing behaviour. For this reason Hyper Typer adopts the opposite approach of presenting players with their familiar Android device keyboard in order to increase the external validity of collected data, even though this approach forces us to relinquish some control over the study setup. Moreover, with our approach measures can be gathered for a variety of text entry mechanisms available on a player's device, not just the one provided by the app.

In order to reach an international audience, phrase sets for transcription in different languages were required. MacKenzie and Soukoreff [22] suggest that phrases for text entry evaluation should be "moderate in length, easy to remember, and representative of the target language". While a number of established, curated phrase sets for text entry evaluation exist [18, 22], the availability of phrase sets in languages other than English is limited. To overcome this limitation, phrase sets in four languages (English, German, French and Spanish) were sampled from transcriptions of the European Parliament using the method described by Leiva et al. [20] for inclusion in the game. Using this method, phrase sets in additional languages can be created and the architecture of the game allows for additional phrase sets to be added remotely and downloaded in the background before a new game is started. The language of the presented phrases is matched to the device language setting, with English chosen as a fallback if no matching phrase set is available.

3.2 Gameplay

Hyper Typer is a futuristic racing game set in a science fiction setting. The game is a single player game which challenges players to compete in an intergalactic spaceship race against computer-controlled opponents. This genre was chosen because racing games are generally a mostly non-violent, yet competitive genre that creates a sense of urgency and motivates players to complete a task as quickly and error-free as possible, which matches the desired objectives for the transcription tasks. The science fiction setting was chosen because it is a common setting in commercial video games



(a) Main Menu **(b) Ship Selection** **(c) Gameplay** **(d) Highscores**

(e) Disclosure **(f) Tutorial** **(g) Global Leaderboard** **(h) Achievements**

Figure 3: Screenshots of Hyper Typer

and provides game designers with considerable artistic freedom in their design as they are not bound by the limits of realism imposed by a realistic contemporary or historical setting.

Upon launching the game for the first time, players are presented with a modal dialogue screen (see Figure 3e) disclosing the game's research purpose, that by playing the game they are participating in a study and about the data collected by the app. After consenting to this disclosure screen, players are automatically presented with a short tutorial (see Figure 3f), providing instructions on how to play the game. This tutorial is only automatically shown when the game is started for the first time to avoid friction for recurring players, but can subsequently be accessed from the main menu in case players need help or instructions at a later time. This tutorial serves a crucial purpose in providing guidance and instructions to first-time players due to the nature of the public distribution and unsupervised usage of the game.

After accepting the disclosure dialogue and finishing the tutorial, players are presented with the main menu of the game (see Figure 3a), where they can start playing a new game, view the highscore list, achievements and global leaderboard, access the tutorial, change game settings or log in and out of Google Play services. Upon subsequent launches of the game after the first launch, players are directly presented with this main menu without further interruption from the disclosure screen or tutorial.

When a player starts a new game, they are first presented with the ship selection screen (see Figure 3b), where they can select one of six space ships. Initially only one ship model is available and additional ship models can be unlocked by completing achievements.

The different ship models only differ in their visual appearance and are purely cosmetic, that is to say, they provide no gameplay benefits or advantages to the player. After selecting a space ship model, the actual gameplay portion of the game begins.

Gameplay is structured around competitive races against computer controlled opponents, where each race consists of four individual stages. A single stage corresponds to the transcription of a single phrase. The game presents the phrase to be transcribed on the top of the screen and the keyboard in the bottom part of the screen (see Figure 3c). In addition, the player's current typing speed in WPM, their position relative to computer-controlled opponents and their overall progress through the stages of a race are shown to provide feedback on a player's performance and progress. Players are challenged to transcribe the presented text as quickly and error-free as possible (the core mechanic of the game) in order to beat their opponents and achieve a highscore. The scoring mechanism takes both speed and error rate into account for calculating a score in order to motivate players to type both quickly and error-free. Players are also explicitly instructed to enter text as quickly and accurately as possible as part of the tutorial of the game.

While Hyper Typer is a single player game, online highscore leaderboards encourage a sense of competition among the player-base. The game keeps two separate highscore lists: a global leaderboard (see Figure 3g), which requires players to use a Google Play account in order to participate, and a local highscore list (see Figure 3d), which is only stored on the player's device, to motivate players without a Google Play account. In addition, the game features achievements implemented via Google Play Services (see Figure 3h) and unlockable ship models (see Figure 3b), as an additional motivational affordance. Unlocking additional ship models is tied to unlocking specific achievements.

3.3 Measures and Data Collection

Before displaying the main menu for the first time, the game informs players about its research purpose and collected data in order to gain informed consent from players. The game collects data about device characteristics, typing performance and player behaviour. A unique installation identifier is created when first starting the app. In addition, the device model, a keyboard identifier, Android SDK version and whether the game was paused are logged for each game round played. No personal information is collected by the game.

The game calculates the following performance measures for text entry speed, error rate metrics and efficiency (for a detailed description see [23]):

- Words per Minute (WPM): Measure of entry rate, based on common assumption that one word consists of 5 characters.
- Adjusted Words per Minute (ADJ.WPM): Adjusted WPM entry rate penalized by uncorrected errors. A penalty exponent of 1.0 is used in our calculation.
- Total Error Rate (TER): Proportion of incorrect-not-fixed and incorrect-fixed characters to the total number of characters entered, equal to the sum of corrected and uncorrected error rates.
- Corrected Error Rate (CER): Proportion of incorrect-fixed characters to the total number of characters entered.
- Uncorrected Error Rate (UER): Proportion of incorrect-not-fixed characters to the total number of characters entered.
- Keystrokes per Character (KSPC): Ratio of the number of entered characters (including backspaces) to the number of characters in the transcribed string.
- Minimum String Distance (MSD): Distance defined as lowest number of error-correction operations required to transform one string into another.
- Correction Efficiency (CE): Ratio of incorrect-fixed characters to the number of keystrokes that correct them.
- Participant Conscientiousness (PC): Proportion of the number of corrected errors to the total number of errors.
- Utilized Bandwidth (UB): Proportion of keystrokes that contribute to the correct aspects of the transcribed string.
- Wasted Bandwidth (WB): Proportion of keystrokes that do not contribute to the correct aspects of the transcribed string, the complement to utilized bandwidth.

Most of these metrics are not exposed to players within the game as they might be confusing for a general audience unfamiliar with the intricacies of text entry evaluation. Only the WPM metric is exposed to players because it is relatively easy to understand and a common feedback mechanism in other typing games. In addition, an overall score calculated by the game that takes both typing speed and uncorrected error rate into account is presented to players. This overall score is also used for ranking in the highscore list and global leaderboard.

In addition to these metrics, the game stores both presented text and transcribed text for each phrase and collects a time-stamped log of all keyboard input events for detailed character-level analysis.

The comprehensive nature of data collection allows the post-hoc calculation of additional text entry performance measures on the server-side. For example, Zhang et al. [44] recently published text entry throughput TET, a performance measure of input efficiency which unifies the speed-accuracy trade off in a single metric. While Hyper Typer originally did not calculate text entry throughput as it was published after the app's release, we were able to use Zhang's procedure to retroactively calculate text entry throughput for our data set.

Collected data is sent in JSON format via REST API and stored in a PostgreSQL database. All data is transferred in the background without player interference. In case of the device being offline, data transmissions are queued in the background and resumed at a later time when connectivity has been restored.

3.4 Development, Testing and Release

Hyper Typer consists of a Unity container embedded in an Android application. In order to speed up development, 3rd party assets from the Unity Asset Store were used. Experiment parameters are remotely configurable through Firebase without requiring the player to update the app. Experiment parameters include the phrase sets, the number of presented phrases per game round, whether the presented text should be in mixed or lower case and whether auto-correction and auto-completion features should be activated or not. The remote configuration can be identical for all participants or partitioned as an A/B test.

Name	Count	Percentage
Samsung Keyboard	155	49.05
Android Default Keyboard	69	21.84
SwiftKey Keyboard	28	8.86
Touchpal Keyboard	13	4.11
Swype for Huawei	11	3.48
LG Keyboard	8	2.53
Others	27	8.54
Total	316	

Table 1: Distribution of text entry mechanisms

	WPM	ADJ.WPM	TET
Mean	34.5	33.7	12.1
SD	12.3	12.0	4.28
	TER	CER	UER
Mean	7.48	5.27	2.21
SD	9.44	8.99	3.08
	PC	UB	WB
Mean	0.547	0.889	0.111
SD	0.434	0.142	0.142

Table 2: Results across all valid transcribed phrases

For the initial release, the game was configured to only present phrases in lowercase letters without special characters and to deactivate auto-correction and auto-completion features for text entry.

Before its public release, Hyper Typer was tested using the Amazon Mechanical Turk platform. Ten participants were recruited to download and play the game five times, with each game round consisting of four stages. No further explanation of the game mechanics was provided in order to gather insights on the comprehensibility of the game design. All participants in the testing phase were able to complete the given task within 18 minutes or less. Five participants provided written feedback which was uniformly positive and indicated no problems of understanding or technical issues. Following the successful testing phase, Hyper Typer was released publicly on July 13th, 2018 on Google Play Store.

4 RESULTS

The following chapter presents the results of an exploratory analysis of the data collected through Hyper Typer over a time span of one year, between its public release on July 13th, 2018 and July 12th, 2019.

4.1 Reach and Participation

Between its release and July 12th, 2019, the game was installed on 2,136 unique devices. Of those installations, 309 distinct installations started the app and played at least one game round, resulting in a total of 1,026 game rounds played, 4,104 phrases transcribed and 99,794 character input events recorded.

It can be assumed that the number of distinct, active installations strongly correlates with the number of players of the game. However this cannot be guaranteed, as one person might have installed the game multiple times or on multiple devices, resulting in multiple installation IDs for a single participant, and one installation of the game might have been shared among several people, conflating multiple players in a single installation ID.

Among those 309 active installations, 3.32 game rounds on average (SD = 10.15) were played. The high standard deviation is the result of a small number of participants playing much more than others, thus causing outliers, with one participant playing 144 game rounds in total.

Highlighting the diversity of Android devices in use today, Hyper Typer was installed and played on 156 different device models. The most popular text entry mechanisms among the playerbase of Hyper Typer were the Samsung Keyboard (155 installations), followed by Android Default Keyboard (69 installations), SwiftKey (28 installations), Touchpal (13 installations) and Swype for Huawei (11 installations), see Table 1. The most common text entry mechanisms were all similar variations of a rather conventional virtual software keyboard design. There was great variety with a total of 27 different text entry mechanisms observed across the playerbase, with 15 of those used by only a single player. The total number of observed text entry mechanisms (n = 316) is slightly larger than the total number of observed installations (n = 309) because some players switched their text entry mechanism in between game rounds.

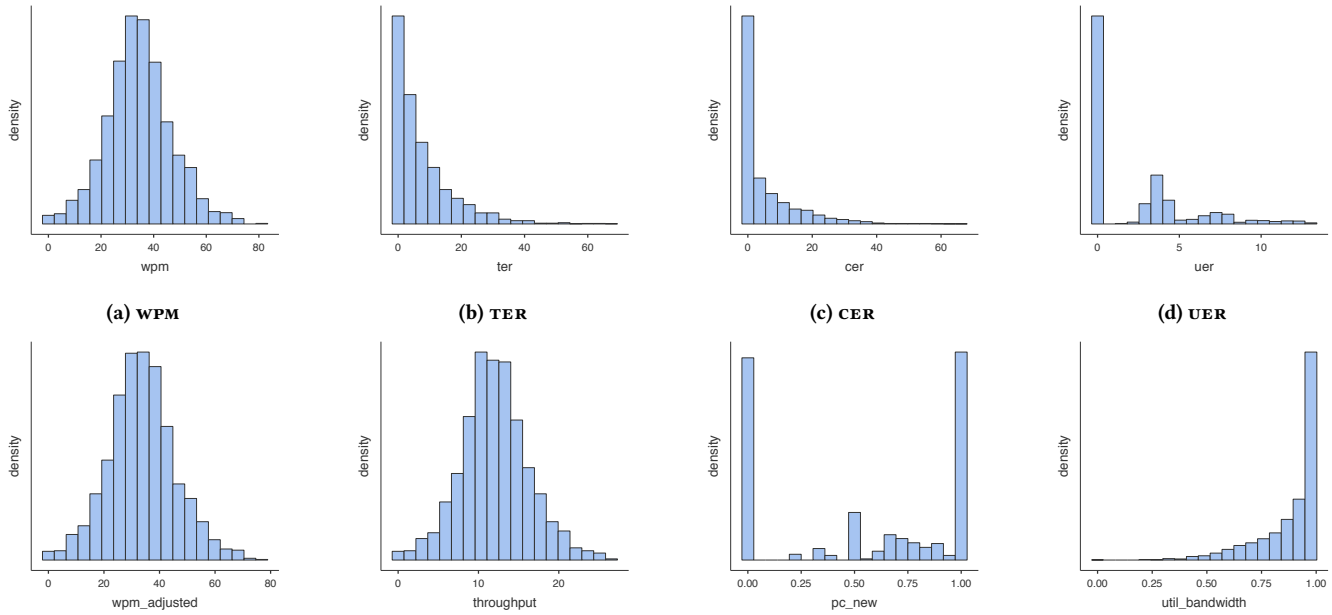
4.2 Data Cleansing

Due to the unsupervised and voluntary nature of gameplay, a large amount of collected data was not usable for subsequent analysis, e.g. because of transcriptions with no discernible relation to the presented phrases or because players paused the game, thus distorting time measurements. Based on thorough manual inspection of the data, appropriate exclusion criteria were defined. To remove garbage input, data from game rounds with a final score of 0 points was excluded, resulting in a total of 615 usable game rounds. In addition, transcribed phrases with an individual score of 0 were also filtered out. 10 game rounds were paused or interrupted and later resumed and therefore also excluded from the results.

This data cleansing results in a total of 2,359 usable transcribed phrases with 71,963 character input events. Of these 2,359 usable transcribed phrases, 1,531 contained at least one error (corrected or uncorrected), whereas 828 contained no error. This final data set after clean up contained data contributed by a total of 73 installations, whereas 236 out of the total 309 active installations contributed no usable data at all.

4.3 Analysis of Overall Text Entry Performance

To gain a comprehensive understanding of real-world text entry performance among the Hyper Typer playerbase, all valid transcribed phrases (n = 2,359) which remained after data cleansing were analyzed. Average text entry speed WPM is 34.5 words per minute (SD = 12.3) and ADJ.WPM 33.7 words per minute (SD = 12.0) when adjusting for uncorrected errors with a penalty exponent of 1.0. Text entry throughput TET is 12.1 (SD = 4.28). In comparison, Castelluci et al. [1] report a text entry rate of 21.4 WPM on



(a) WPM (b) TER (c) CER (d) UER

(e) ADJ.WPM (f) TET (g) PC (h) UB

Figure 4: Distribution of collected measures across all valid transcribed phrases

a default Android keyboard . In a later study Castellucci et al. [2] report text entry rates of 20.9 WPM for one-handed and 20.8 WPM for two-handed input on a default Android keyboard. These numbers are distinctly lower than our results. Reyal et al. [30] evaluated text entry rates across five sessions for typing on the Google Gboard keyboard in a lab experiment and an ESM study, reporting text entry rates of 29.1 WPM (session 1) and 32.8 WPM (session 5) for the lab experiment condition and 30.1 WPM (session 1) and 31.1 WPM (session 5) for the ESM study condition. While still lower than our results, these numbers are closer to what we observed. Zhang et al. [44] report text entry rates of 51.91 WPM for normal, 40.91 WPM for accurate and 60.16 WPM for fast typing behaviour on the Google Gboard keyboard with auto-correction and auto-completion features turned off. These results are higher than our findings.

Average error rates are total error rate TER 7.48 % (SD = 9.44), corrected error rate CER 5.27 % (SD = 8.99) and uncorrected error rate UER 2.21 % (SD = 3.08). In comparison, Castelluci et al. [1] report a total error rate TER of 11.8 % on a default Android keyboard . In a later study Castellucci et al. [2] report total error rates TER of 7.1 % for one-handed and 13.8 % for two-handed input on a default Android keyboard. Zhang et al. [44] report uncorrected error rates UER of 6.1 % for normal, 0.8 % for accurate and 12.2 % for fast typing behaviour on the Google Gboard keyboard with auto-correction and auto-completion features turned off.

Taking into account only phrase transcriptions were an error occurred (1,531), average participant conscientiousness PC is 0.547 (SD = 0.434). Average utilized bandwidth UB is 0.889 (SD = 0.142), with wasted bandwidth WB its complement 0.111 (SD = 0.142).

Figure 4 illustrates the distribution of the collected measures across all valid transcribed phrases. The distribution of participant conscientiousness PC (see Figure 4g) is noteworthy with clusters at the extremes of the distribution. More specifically, of the 1,531 phrases which contained errors, for 539 phrases (35.2 %) all errors were corrected, for 527 phrases (34.4 %) no errors were corrected and 465 phrases (30.4 %) were partially corrected. This suggests that players adopted one of three possible error correction strategies in similar proportions: either to correct all errors in a phrase, to correct some errors, or to correct no errors at all.

To assess the overall efficiency of the three different error correction strategies, we compared text entry throughput TET. For this analysis the data set is split in three groups: accurate (group A) for phrases where all errors were corrected, fast (group F) where no errors were corrected and neutral (group N) where some errors were corrected. Phrases entered without error are filtered for this analysis as no deductions regarding a particular correction strategy can be made. A comparison between the three groups showed a significant effect of correction strategy on text entry throughput TET ($F(2, 989) = 92.1$, $p < 0.001$). Group F had the highest TET ($M = 12.88$, $SD = 4.13$), followed by group A ($M = 10.70$, $SD = 3.51$) and group N ($M = 9.54$, $SD = 3.64$). Pairwise comparisons using a Games-Howell post-hoc test revealed a statistically significant difference between all groups (A - N: $p < 0.001$; A - F: $p < 0.001$; F - N: $p < 0.001$). This is in line with results by Zhang et al. [44], who also found that an accuracy-focused cognitive disposition results in lower text entry throughput on smartphones, but not in desktop computing. These results indicate that correcting errors is comparatively less efficient than ignoring errors in mobile text entry.

4.4 Analysis by Keyboard

A comparison between the three most-used keyboards among the playerbase (Android Default Keyboard, Samsung Keyboard, SwiftKey Keyboard) of the game showed a significant effect of keyboard type on text entry speed WPM ($F(2,263) = 23.33$, $p < 0.001$) as well as total error rate TER ($F(2,264) = 13.97$, $p < 0.001$), corrected error rate CER ($F(2,267) = 4.68$, $p = 0.010$) and uncorrected error rate UER ($F(2, 248) = 31.56$, $p < 0.001$), see Table 3).

Players using the Android Default Keyboard had the fastest text entry speed WPM ($M = 37.00$, $SD = 11.18$) compared to SwiftKey ($M = 34.50$, $SD = 11.44$) and the Samsung Keyboard ($M = 31.63$, $SD = 11.50$). Pairwise comparisons using a Games-Howell post-hoc test revealed a statistically significant difference between the Android Default Keyboard and the Samsung Keyboard ($p < 0.001$), but no significant difference between Android Default Keyboard and SwiftKey, and between SwiftKey and the Samsung Keyboard.

The Android Default Keyboard exhibited the lowest total error rate TER, corrected error rate CER and uncorrected error rate UER (TER: $M = 6.11$, $SD = 8.98$; CER: $M = 4.45$, $SD = 8.56$; UER: $M = 1.66$, $SD = 2.84$) compared to SwiftKey (TER: $M = 8.51$, $SD = 8.20$; CER: $M = 4.53$, $SD = 7.32$; UER: $M = 3.98$, $SD = 3.69$) and the Samsung Keyboard (TER: $M = 9.48$, $SD = 10.64$; CER: $M = 6.72$, $SD = 10.81$; UER: $M = 2.76$, $SD = 3.52$). Pairwise comparisons of total error rate TER using a Games-Howell post-hoc test revealed statistically significant differences between the Android Default Keyboard and the Samsung Keyboard ($p < 0.001$) as well as between the Android Default Keyboard and SwiftKey ($p = 0.006$), but no significant difference between SwiftKey and the Samsung Keyboard. Pairwise comparisons of corrected error rate CER revealed a statistically significant difference between the Android Default Keyboard and the Samsung Keyboard ($p = 0.007$), but no others. Pairwise comparisons of uncorrected error rate UER revealed statistically significant differences between all three keyboards (Android Default - Samsung: $p < 0.001$; Android Default - SwiftKey: $p < 0.001$; Samsung - SwiftKey: $p = 0.008$).

These results show the Android Default Keyboard performing best regarding both speed and accuracy. However, it must be pointed out that users of the Android Default Keyboard were much more active players compared to users of the other two keyboards. Taking into account players who contributed at least one valid transcription, players using the Android Default Keyboard contributed an average of 34.3 valid transcribed phrases compared to 15.6 valid transcribed phrases per player for the Samsung Keyboard and 12.6 valid transcribed phrases per player for SwiftKey Keyboard. As such, one possible explanation for performance differences between keyboards is that players using the Android Default Keyboard were more strongly motivated by the game, had more practice and thus performed better.

4.5 Analysis by Language

To examine the effect of phrase set language on text entry speed and error rates, we compared the English (n = 1,225) and German (n = 1,115) language phrase sets. The French (n = 0) and Spanish (n = 19) language phrase sets were excluded because they didn't produce sufficient data for analysis.

A Welch's unequal variances t-test found a significant effect of phrase set language on text entry speed WPM ($t(2,289) = 16.25$, $p < 0.001$), with WPM being significantly higher in German ($M = 38.71$, $SD = 11.89$) than English ($M = 30.91$, $SD = 11.28$).

	Android Default	Samsung	SwiftKey	p
n	1,339	235	126	
WPM	37.00 (11.1)	31.63 (11.50)	34.50 (11.44)	<0.001
TER	6.11 (8.98)	9.48 (10.64)	8.51 (8.20)	<0.001
CER	4.45 (8.56)	6.72 (10.81)	4.53 (7.32)	=0.010
UER	1.66 (2.84)	2.76 (3.52)	3.98 (3.69)	<0.001

Table 3: Performance by keyboard

	English	German	p
n	1,225	1,115	
WPM	30.91 (11.28)	38.71 (11.89)	<0.001
TER	7.59 (9.00)	7.29 (9.88)	=0.454
CER	5.26 (8.42)	5.24 (9.54)	=0.939
UER	2.32 (3.02)	2.06 (3.13)	=0.038

Table 4: Performance by phrase set language

A comparison of error rates using Welch's unequal variances t-test found a significant difference between English ($M = 2.32$, $SD = 3.02$) and German ($M = 2.06$, $SD = 3.13$) regarding uncorrected error rate ($t(2,298) = -2.08$, $p = 0.038$), but no significant effect of language on total error rate TER and corrected error rate CER (see Table 4).

The notable difference in text entry speed between English and German language phrase sets warrants further consideration. One possible cause is an inherent bias in phrase sets of different languages, caused by the different balance of words per phrase and letters per word [7]. Another possible explanation is that there's a higher proportion of non-native English speakers among players presented with the English phrase set as this phrase set is used as a fallback if no phrase set in a matching language for the device language setting is available. Isokoski and Linden [15] found that text entry in a foreign language had a detrimental effect on both speed and error rate.

5 DISCUSSION

The results demonstrate the feasibility of measuring text entry performance with a serious game through public distribution on Google Play Store. We believe that the data obtained through Hyper Typer is a good representation of realistic and natural typing behaviour, as it resulted from volunteer participants entering text on their own, personal device with their familiar text entry method and preferences intact.

That being said, the collection of data without direct instructions and supervision from researchers has its disadvantages where internal validity is concerned. Researchers have limited control over how, when and where people participate. Participants might be distracted or interrupted, operate the game in unforeseen ways or misunderstand the instructions. Certain aspects of the study conditions, such as the device and keyboard used for text entry, hand

position (e.g. one or two handed, left or right handed), participant activity or context of use were outside our control.

In order to ensure the quality and integrity of the collected data, certain collected usage characteristics such as device identifier, keyboard type or pauses in gameplay can be used to reject inapplicable or invalid results on a case by case basis. Beyond the usage characteristics collected by Hyper Typer, additional sensor data such as accelerometer data could be used to infer additional details about usage context.

In addition, researchers have limited influence on sampling and thus participants might not be representative of the general population or a specific intended target audience. One particular concern regarding the use of serious games for research purposes is that they specifically target participants who are interested in playing video games, whereas those parts of the population with no interest in video games are largely excluded. This inherent bias must be carefully considered by researchers.

The effort required for successful public release of a research app must not be underestimated by researchers interested in the endeavour. The robustness and level of polish required for an app intended for public release and distribution is considerably higher compared to a research prototype intended for internal use in a controlled study environment. One specific challenge when releasing an Android app is the wide variety of different Android devices on the market today, e.g. Hyper Typer was installed and played on 156 different Android device models. This is an enormous challenge for device compatibility testing. For example, despite comprehensive testing on different devices before public release, we only became aware of a game-breaking layout bug related to certain screen aspect ratios of some Android devices months after release. Once aware of the bug we were able to fix it by releasing an update to the app. However, this kind of bug is not only frustrating for interested players and limiting the game's potential audience, but also carries the risk of distorting the validity of collected data by excluding certain groups from participating in the study.

Reaching an audience for the game has proven more challenging than anticipated. Despite our best efforts to make the game known through word of mouth and online advertising efforts, the game never reached as many people as we hoped for and anticipated. This issue could probably be remedied with more experience in successful app marketing or more substantial advertising spending. Furthermore, converting people who download the app into active participants is another challenge. Of 2,136 total downloads only 309 played at least one game round, and only 73 installations (3.41 % of total installations) contributed usable data to our results. Researchers interested in running public studies on app stores must be aware that simply building an app and making it publicly available is no guarantee for reaching an audience and should plan their marketing and on-boarding activities from the outset.

The amount of data collected through Hyper Typer is relatively large compared to some other lab-based experiments for mobile text entry evaluation, but relatively small compared to other successful deployments of serious research games on app stores (99,794 keyboard input events compared to e.g. 120,626,225 touch events in [12] or 47,770,625 keystrokes in [13]). However, these earlier studies are several years in the past and the number of apps available on Google Play Store has massively increased since then, from ca. 100,000 apps in October 2010 to ca. 2.7 million apps in June 2019 [36]. This increased competition is a possible explanation for our difficulties in finding a larger audience, as it is more difficult to stand out on the crowded storefronts of app stores today compared to several years ago.

The amount of invalid data collected by the game warrants further investigation. This could be a result of insufficient instructions, poor usability or disinterest of participants. In addition, specific game design choices such as incentive structures and scoring mechanisms could distort the typing behaviour of players. To better understand the effect of these issues we plan a comparative study of data collected from unsupervised players and from a controlled lab experiment in the future.

Our analysis of text entry performance based on data collected through Hyper Typer is higher than the performance reported in some earlier lab-based experiments [1, 2, 30], sometimes considerably so. There are a number of possible explanations for this: Firstly, it is possible that the familiarity with their own device positively impacted players' text entry performance. Secondly, it is possible that the game more strongly motivated and incentivized players to perform at their best compared to an artifical lab-based task. Lastly, given that these earlier experiments are now several years in the past, it is possible that advances in smartphone hardware and software had a positive effect on text entry performance or that people in general have become more adept at entering text using virtual keyboards on a smartphone.

6 CONCLUSION

In this paper we present the design process and implementation details of our serious research game Hyper Typer and provide specific recommendations and requirements for text entry evaluation in a serious game. The game successfully fulfills its purpose of gathering comprehensive data about text entry performance and behaviour from players for exploratory analysis. Following the public release of the game, 4,104 transcribed phrases and 99,794 keyboard input events from 309 active installations were collected over a one year time span. We present the results of our exploratory analysis after data cleansing, which reduced our data set to 2,359 valid transcribed phrases and 71,963 keyboard input events. We reflect on the experience and challenges in publicly releasing a serious research game on app stores. In retrospect we consider our approach of using a serious game to collect research data successful, but acknowledge that additional efforts to reach a wider audience are required for this approach to reach its full potential.

Hyper Typer remains publicly available [32] and further dissemination, data collection and exploratory data analysis are ongoing efforts. Furthermore, the configuration options provided by the app allow its use for exploring more focused research questions, such as the effects of auto-correction and auto-completion features, the influence of screen size on text entry performance or a comparison of different phrase sets. In addition, further internationalization, especially beyond Latin-script alphabets, could extend the reach and applicability of the game and would open up new research questions for examination. Finally, in the spirit of open research the raw data for this analysis is publicly available [34].

REFERENCES

[1] Steven J. Castellucci and I. Scott Mackenzie. 2011. Gathering text entry metrics on android devices. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems*. 1507–1512. https://doi.org/10.1145/1979742.1979799

[2] Steven J. Castellucci and I. Scott Mackenzie. 2013. Gathering Text Entry Metrics on Android Devices. In *Proceedings of the International Conference on Multimedia and Human-Computer Interaction - MHCI 2013*. 1–7.

[3] Corey Clark, Ira Greenberg, and Myque Ouellette. 2018. A model for integrating human computing into commercial video games. *2018 IEEE 6th International Conference on Serious Games and Applications for Health, SeGAH 2018* (2018), 1–8. https://doi.org/10.1109/SeGAH.2018.8401316

[4] Damien Djaouti, Julian Alvarez, and Jean-Pierre Jessel. 2011. Classifying serious games: the G/P/S model. In *Handbook of research on improving learning and motivation through educational games: Multidisciplinary approaches*. 118–136. https://doi.org/10.4324/9780203891650 arXiv:arXiv:1011.1669v3

[5] Exideas. 2018. MessagEase Keyboard. Retrieved August 16, 2019 from https://www.exideas.com/ME/.

[6] Leah Findlater, Joan Zhang, Jon E. Froehlich, and Karyn Moffatt. 2017. Differences in Crowdsourced vs. Lab-based Mobile and Desktop Input Performance Data. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. 6813–6824. https://doi.org/10.1145/3025453.3025820

[7] Marc Franco-Salvador and Luis A. Leiva. 2018. Multilingual phrase sampling for text entry evaluations. *International Journal of Human Computer Studies* 113, February 2017 (2018), 15–31. https://doi.org/10.1016/j.ijhcs.2018.01.006

[8] Niels Henze. 2012. Hit it!: an apparatus for upscaling mobile HCI studies. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*. 1333–1338. https://doi.org/10.1145/2212776.2212450

[9] Niels Henze and Martin Pielot. 2013. App stores: external validity for mobile HCI. *Interactions* 20, 2 (2013), 33–38. https://doi.org/10.1145/2427076.2427084

[10] Niels Henze, Martin Pielot, Benjamin Poppinga, Torben Schinke, and Susanne Boll. 2011. My app is an experiment: Experience from user studies in mobile app stores. *International Journal of Mobile Human Computer Interaction (IJMHCI)* 3, 4 (2011), 71–91.

[11] Niels Henze, Benjamin Poppinga, and Susanne Boll. 2010. Experiments in the wild: Public evaluation of off-screen visualizations in the Android Market. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*. 675–678. https://doi.org/10.1145/1868914.1869002

[12] Niels Henze, Enrico Rukzio, and Susanne Boll. 2011. 100,000,000 Taps: Analysis and Improvement of Touch Performance in the Large. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. 133–142.

[13] Niels Henze, Enrico Rukzio, and Susanne Boll. 2012. Observational and experimental investigation of typing behaviour using virtual keyboards for mobile devices. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*. 2659–2668. https://doi.org/10.1145/2207676.2208658

[14] Robin Hunicke, Marc LeBlanc, and Robert Zubek. 2004. MDA: A Formal Approach to Game Design and Game Research. In *Workshop on Challenges in Game AI*. 1–4. https://doi.org/10.1.1.79.4561

[15] Poika Isokoski and Timo Linden. 2004. Effect of foreign language on text transcription performance: Finns writing English. In *Proceedings of the third Nordic conference on Human-computer interaction*. 109–112. https://doi.org/10.1145/1028014.1028032

[16] Thomas Költringer and Thomas Grechenig. 2004. Comparing the Immediate Usability of Graffiti 2 and Virtual Keyboard. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems*. 1175–1178. https://doi.org/10.1145/985921.986017

[17] Steven Komarov, Katharina Reinecke, and Krzysztof Z. Gajos. 2013. Crowdsourcing performance evaluations of user interfaces. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 207–216. https://doi.org/10.1145/2470654.2470684

[18] Per Ola Kristensson and Keith Vertanen. 2012. Performance comparisons of phrase sets and presentation styles for text entry evaluations. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*. 29–32. https://doi.org/10.1145/2166966.2166972

[19] Per Ola Kristensson and Shumin Zhai. 2007. Learning shape writing by game playing. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems*. 1971–1976. https://doi.org/10.1145/1240866.1240934

[20] Luis A. Leiva and Germán Sanchis-Trilles. 2014. Representatively memorable: sampling the right phrase set to get the text entry experiment right. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, Toronto, Ontario, Canada, 1709–1712. https://doi.org/10.1145/2556288.2557024

[21] I. Scott MacKenzie and R. William Soukoreff. 2002. Text Entry for Mobile Computing: Models and Methods,Theory and Practice. *Human-Computer Interaction* 17, 2-3 (2002), 147–198. https://doi.org/10.1080/07370024.2002.9667313

[22] I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase sets for evaluating text entry techniques. In *CHI'03 extended abstracts on Human factors in computing systems*. 754–755. https://doi.org/10.1145/765968.765971

[23] I. Scott MacKenzie and Kumiko Tanaka-Ishii. 2010. *Text entry systems: Mobility, accessibility, universality*. Elsevier.

[24] Donald McMillan, Alistair Morrison, Owain Brown, Malcolm Hall, and Matthew Chalmers. 2010. Further into the wild: Running worldwide trials of mobile systems. In *International Conference on Pervasive Computing*, Vol. 6030 LNCS. 210–227. https://doi.org/10.1007/978-3-642-12654-3_13

[25] Donald McMillan, Alistair Morrison, and Matthew Chalmers. 2011. A Comparison of Distribution Channels for Large-Scale Deployments of iOS Applications. *International Journal of Mobile Human Computer Interaction (IJMHCI)* 3, 4 (2011), 1–17. https://doi.org/10.4018/jmhci.2011100101

[26] Donald McMillan, Alistair Morrison, and Matthew Chalmers. 2013. Categorised ethical guidelines for large scale mobile HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. 1853. https://doi.org/10.1145/2470654.2466245

[27] David R. Michael and Sandra L. Chen. 2005. *Serious Games: Games That Educate, Train, and Inform*. Muska & Lipman/Premier-Trade.

[28] Alistair Morrison, Donald McMillan, and Matthew Chalmers. 2014. Improving consent in large scale mobile HCI through personalised representations of data. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction Fun, Fast, Foundational - NordiCHI '14*. 471–480. https://doi.org/10.1145/2639189.2639239

[29] Ondrej Polacek, Adam J Sporka, and Brandon Butler. 2013. Improving the methodology of text entry experiments. In *Cognitive Infocommunications (CogInfoCom), 2013 IEEE 4th International Conference on*. IEEE, 155–160.

[30] Shyam Reyal, Shumin Zhai, and Per Ola Kristensson. 2015. 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*. 679–688. https://doi.org/10.1145/2702123.2702597

[31] Dmitry Rudchenko, Tim Paek, and Eric Badger. 2011. Text text revolution: A game that improves text entry on mobile touchscreen keyboards. In *International Conference on Pervasive Computing.*, Vol. 6696 LNCS. 206–213. https://doi.org/10.1007/978-3-642-21726-5_13

[32] Richard Schlögl, Christoph Wimmer, and Thomas Grechenig. 2018. Hyper Typer. Google Play Store. Retrieved August 16, 2019 from https://play.google.com/store/apps/details?id=at.hypertyper.

[33] Richard Schlögl, Christoph Wimmer, and Thomas Grechenig. 2019. Hyper Typer: A Serious Game for Measuring Mobile Text Entry Performance in the Wild. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, LBW0259.

[34] Richard Schlögl, Christoph Wimmer, and Thomas Grechenig. 2019. Hyper Typer Project Website. Website. Retrieved August 16, 2019 from https://deco.inso.tuwien.ac.at/hypertyper/.

[35] R. William Soukoreff and I. Scott MacKenzie. 2003. Metrics for text entry research-an evaluation of MSD and KSPC, and a new unified error metric. In *Proceedings of the conference on Human factors in computing systems - CHI '03*, Vol. 5. 113–120. https://doi.org/10.1145/642611.642632

[36] Statista. 2019. Number of available applications in the Google Play Store from December 2009 to June 2019. Website. Retrieved August 16, 2019 from https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

[37] Dominic Szablewski, Sebastian Gerhard, and Andreas Lösch. 2011. ZType. Website. Retrieved August 16, 2019 from https://zty.pe/.

[38] Authors unknown. 2008. TypeRacer. Website. Retrieved August 16, 2019 from https://play.typeracer.com/.

[39] Authors unknown. 2010. 8pen. Website. Retrieved August 16, 2019 from http://www.8pen.com/.

[40] Keith Vertanen, Justin Emge, Haythem Memmi, and Per Ola Kristensson. 2014. Text Blaster: A Multi-Player Touchscreen Typing Game. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*. 379–382. https://doi.org/10.1145/2559206.2574802

[41] Luis von Ahn and Laura Dabbish. 2008. Designing Games With a Purpose. *Commun. ACM* 51, 8 (2008), 58–67. https://doi.org/10.1145/1378704.1378719

[42] Whirlscape. 2013. Minuum Keyboard. Website. Retrieved August 16, 2019 from http://minuum.com/.

[43] Jacob O. Wobbrock and Brad A. Myers. 2006. Analyzing the input stream for character- level errors in unconstrained text entry evaluations. *ACM Transactions on Computer-Human Interaction* 13, 4 (2006), 458–489. https://doi.org/10.1145/1188816.1188819

[44] Mingrui "Ray" Zhang, Shumin Zhai, and Jacob O. Wobbrock. 2019. Text Entry Throughput: Towards Unifying Speed and Accuracy in a Single Performance Metric. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 636.