Text Entry Throughput

Towards Unifying Speed and Accuracy in a Single Performance Metric

Mingrui "Ray" Zhang
The Information School
DUB Group
University of Washington
Seattle, WA, USA 98195
mingrui@uw.edu

Shumin Zhai
Google, Inc.
1600 Amphitheater Parkway
Mountain View, CA, USA 94043
zhai@google.com

Jacob O. Wobbrock
The Information School
DUB Group
University of Washington
Seattle, WA, USA 98195
wobbrock@uw.edu

ABSTRACT

Human-computer input performance inherently involves speed-accuracy tradeoffs—the faster users act, the more inaccurate those actions are. Therefore, comparing speeds and accuracies separately can result in ambiguous outcomes: Does a fast but inaccurate technique perform better or worse overall than a slow but accurate one? For pointing, speed and accuracy has been unified for over 60 years as *throughput* (bits/s) (Crossman 1957, Welford 1968), but to date, no similar metric has been established for text entry. In this paper, we introduce a text entry method-independent throughput metric based on Shannon information theory (1948). To explore the practical usability of the metric, we conducted an experiment in which 16 participants typed with a laptop keyboard using different cognitive sets, *i.e.*, speed-accuracy biases. Our results show that as a performance metric, text entry throughput remains relatively stable under different speed-accuracy conditions. We also evaluated a smartphone keyboard with 12 participants, finding that throughput varied least compared to other text entry metrics. This work allows researchers to characterize text entry performance with a single unified measure of input efficiency.

CCS CONCEPTS

• Human-centered computing → **Text input; HCI theory; concepts and models.**

KEYWORDS

Text entry; text input; throughput; information theory; human performance; speed; accuracy; speed-accuracy tradeoff; efficiency.

ACM Reference format:

Mingrui "Ray" Zhang, Shumin Zhai, Jacob O. Wobbrock. 2019. Text Entry Throughput: Towards Unifying Speed and Accuracy in a Single Performance Metric. In *2019 CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland, UK.* ACM, New York, NY, USA. 13 pages. https://doi.org/10.1145/3290605.3300866

CHI 2019, May 4–9, 2019, Glasgow, Scotland UK

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
DOI: https://doi.org/10.1145/3290605.3300866

1 INTRODUCTION

Since the earliest days of interactive computing, text entry has remained one of the most important and frequent of all computing activities performed by humans. Alongside pointing, text entry is fundamental to interactive computing systems—almost all platforms provide it [13], and immense energy has been poured into creating advanced methods (*e.g.*, [34]), calculating performance bounds (*e.g.*, [23]), improving evaluation methods (*e.g.*, [25]), and providing evaluation tools (*e.g.*, [1,17,31]). When assessing the performance of text entry methods, however, a fundamental challenge remains: As with all human performance, speed and accuracy trade off against each other, complicating the assessment of text entry methods in the presence of such tradeoffs. How can we draw firm performance conclusions in the presence of such tradeoffs? Table 1 illustrates the problem:

Table 1. Two hypothetical text entry methods posing a speed-accuracy tradeoff. Which has better overall performance? Throughput gives a way of equitably comparing methods, even across studies.

Method	Speed (WPM)	Error Rate (%)
A	13.2	4.8
B	16.7	7.2
Difference (B – A)	*3.5 (B is better)*	*2.4 (B is worse)*

Modern text entry evaluations (see, *e.g.*, [13,25,30,31]) allow for the separate measurement of speed and accuracy, usually reported, respectively, as words per minute (WPM) [11] and three error rates: uncorrected, corrected, and total errors [25]. Strings are presented for transcription and participants are instructed to "proceed quickly and accurately" [25,29,32]. Uncorrected errors are those that remain in the transcribed string, and are therefore at odds with speed. Corrected errors are those made but fixed *during* entry (*e.g.*, backspaced)—as such, they take time, and are therefore subsumed in the speed measure. Our question, then, is how to reconcile speed with *uncorrected* errors.

The instruction to "proceed quickly and accurately" is an interesting, and concerning, one [33]. Every human actor has his or her own internal subjective speed-accuracy bias, which may change with purpose and context. Thus, separate measures of speed and accuracy will vary under different speed-accuracy conditions. Our goal is to devise, theoretically and empirically, a robust performance measure for text entry that conveys the information found in speed and accuracy measures, while also remaining stable across various speed-accuracy biases.

A text input system is, quite literally and conceptually, a communications channel in the information-theoretic sense. Shannon's information theory [19] should therefore shed light on the evaluation of text input methods. Intuitively, the amount of information transmitted via a text input method per unit time, termed *throughput*, reflects the input efficiency of the method.

We propose that text entry evaluations are describable in terms of Shannon's model. In a text entry transcription task, the presented string (*P*) can be regarded as the information source; the transcribed string (*T*) can be regarded as the information destination; and the system of human-plus-input-method can be regarded as a discrete channel perturbed by noise, which are errors. Characters are signals transmitted through the input process (*e.g.*, typing), modified by noise and displayed on the screen. From such an information transmission model, we derived a formula to calculate throughput based on text entry speed and the uncorrected error rate. Our experiment showed that for the same person with the same text entry method, our throughput measure exhibited less variation compared to other text entry metrics across different speed-accuracy conditions, suggesting that our measure characterizes the communications channel itself, apart from a human actor's particular speed-accuracy bias.

The contributions of this work are: (1) The formalization of text entry transcription tasks as Shannon information transmission tasks; (2) A new throughput metric unifying speed and accuracy in text entry, which enables comparisons of overall text entry efficiency; and (3) Empirical results validating the stability of this new throughput metric across different speed-accuracy biases, thereby characterizing a text entry method's communication efficiency. Although we do not seek to replace speed and accuracy as text input performance metrics, and we encourage them to be reported in all future studies, we believe that our new throughput metric can provide a valuable unified measure for characterizing the overall efficiency of text input methods.

2 RELATED WORK

In the following two subsections, we first introduce speed-accuracy normalization in aimed pointing movements, which provides a precedent for our work in text entry. Second, we describe efforts to combine speed and accuracy in text entry specifically.

2.1 Speed-Accuracy Normalization in Aimed Pointing Movements

Related research on the speed-accuracy tradeoff has a long history, specifically arising with Fitts' law [3] in 1954 for aimed pointing movements. In 1969, Fitts' colleague Pew introduced the "speed-accuracy operating characteristic," noting that "the relationship between speed and accuracy of performance under a wide variety of task conditions reveals a linear relationship between log odds in favor of a correct response and reaction time" [18] (p. 16).

In aimed pointing specifically, the speed-accuracy tradeoff is clear: if one moves faster, the spread-of-hits around a target will be greater than if one moves more slowly, as shown in Figure 1.

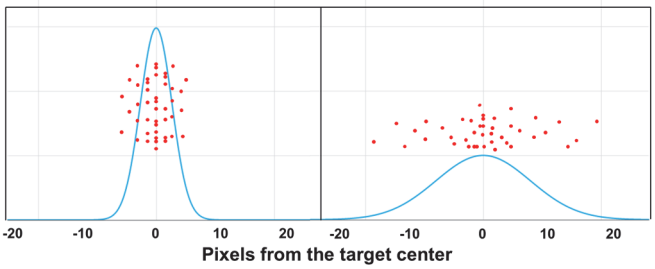

Figure 1. The speed-accuracy tradeoff in aimed pointing. (*left*) Slower movements. (*right*) Faster movements.

Fitts' law [3] is an empirical model that combines the spread-of-hits and pointing time. Inspired by Shannon information theory [19], the law predicts the movement time of an aimed pointing task: It regards the human motor system as a communications channel, which transmits information during pointing.

Based on Fitts' law, Crossman [2] provided a correction to normalize the speed-accuracy tradeoff with his corrected throughput measure, which was later popularized by Welford [28] (pp. 147-149). Throughput, whose units is bits per second (bits/s), characterizes an aimed pointing task for its performance efficiency. For the mathematical details of calculating throughput, see MacKenzie, who gives a nice overview [10] (pp. 106-109).

To evaluate the effectiveness of throughput, Mackenzie and Isokoski [12] conducted a series of pointing tasks under different speed-accuracy conditions, and found that throughput was indeed independent of the specific speed-accuracy tradeoff being made. We should want for a similarly "stable" throughput metric for text entry. Such

metrics allow for cross-study comparisons, because unlike speed and accuracy alone, which depend on task conditions (*e.g.*, target size and distance in aimed pointing), throughput is independent of these.

2.2 Speed-Accuracy Tradeoffs in Text Entry

Previous efforts to deal with the speed-accuracy tradeoff in text entry include prohibiting text entry errors outright—for example, when wrong characters are entered, they are simply prevented from appearing on-screen, thereby only measuring speed as a single "unified" metric [26]. Other studies disabled error correction [16]. Yet other studies simply ignored errors [8]. Such artificial procedures to produce a single speed-accuracy measure do not yield a true unified metric, and they overly constrain text entry evaluations, making them highly artificial.

As another simple unified metric, Wobbrock [30] proposed *Adjusted Words per Minute (AdjWPM)* as:

$$AdjWPM = WPM \times (1 - E) \quad (1)$$

where E refers to the uncorrected error rate [25]. However, the definition of *AdjWPM* lacks any theoretical basis. For example, why should the penalty for uncorrected errors on WPM be linear? Why should a 5% uncorrected error rate result in the raw WPM being reduced to 95% of its original value? In any case, we compare our proposed throughput metric to AdjWPM in our experiment, as AdjWPM is one of the few existing metrics to mathematically combine speed and accuracy.

Sourkoreff & Mackenzie [25] defined *Utilized Bandwidth* and *Wasted Bandwidth* to represent the amount of useful information transferred during text entry. However, both metrics reflect percentages of correct keystrokes and do not take *time* into account. Thus, they are not unified speed-accuracy metrics.

Soukoreff, in his doctoral dissertation [21], proposed a model with a goal similar to that of this paper. However, Soukoreff's model only considers correct characters in the transcribed string, and calculates throughput in terms of the information contained in the entry rate of correct characters. This calculation is both theoretically and practically insufficient. Let us consider an example to illustrate the problems of only considering correct characters. Two people attempt to type "*abcde*" in the same amount of time; the first types "*abcde*" and the second types "*aabbccddee*", with every letter doubled. Their throughput under Sokoreff's model would be the same, because they both contain the same correct characters. However, the second typist typed many more characters. Clearly, more than just correct characters must be considered in any complete throughput calculation.

3 DESIRED PROPERTIES OF A THROUGHPUT METRIC FOR TEXT ENTRY

Below, we convey our view of the desired properties of an ideal text entry throughput metric. These properties generally hold for other text entry metrics [30] and for throughput in Fitts' law [12]:

1. ***Method independence.*** Our metric should be text entry method-independent. As with metrics like words per minute [11] and uncorrected error rate [25], our throughput metric should not require method-specific knowledge to be calculated. In other words, throughput should apply to any text entry method.
2. ***Use of P, T, and time only.*** Following method-independence, our throughput metric should be computed only with knowledge of the presented string (P), transcribed string (T), and total entry time. Everything that happens *during* the entry process, such as corrections, should be unrelated to the metric.
3. ***Performance isolation.*** The calculation of the metric should only depend on the performance of the system (human and input method), like existing speed [11] and accuracy [25] metrics. It should not include external factors such as which phrase set is used in an evaluation. Although the test phrases do influence the metric, they should represent the intended application domain. Otherwise, an experimenter could "cheat" by selecting a phrase set favorable to the method.
4. ***Speed-accuracy invariance.*** Within reasonable limits, the metric should be relatively stable across participants' different subjective speed-accuracy biases, thereby characterizing the communications channel, not a particular participant bias.
5. ***Compliance with established text entry evaluation procedures.*** Evaluation procedures should not be artificially constrained to enable the use of our new throughput metric. Researchers should be able to calculate the metric without making participants enter text in a specific way, *e.g.*, by forcing them to enter every character correctly.
6. ***Simplicity of measurement.*** Ideally, the metric should be easy to measure. By "easy," we mean both the data collection and calculation steps should be as straightforward as possible, similar to calculating existing text entry error rates [25].

4 MAPPING SHANNON INFORMATION-THEORY TO TEXT ENTRY

Before we can define throughput for text entry, we must establish how Shannon's information-theoretic communications model [19] maps to the text entry

transcription process. In Shannon's original model of a communications system (Figure 2), a message is produced by an *information source*, transmitted by a *transmitter* through a *channel*, received by a *receiver*, and recorded at a *destination*. The message can be perturbed by *noise*, meaning the message sent is not necessarily the message received.

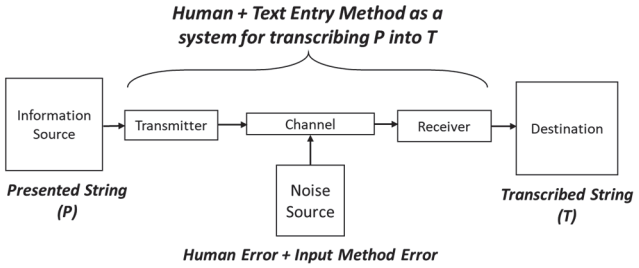

Figure 2. Shannon's information transmission model, with labels in bold mapping the model to the text entry transcription process.

In a text entry transcription task, after the presented string (*P*) is first shown, the participant enters the transcribed string (*T*) using whatever text entry method is under investigation. Usually, a generic test-bed application is used, which presents phrases from a representative corpus and provides for the transcribed text to be entered directly below, as shown in Figure 3.

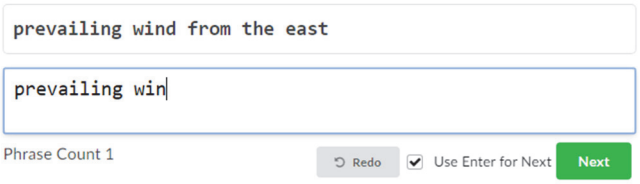

Figure 3. A text entry transcription task in the *TextTest++* evaluation tool presented in this paper.

The presented string (*P*) can be viewed as the *information source*, and the transcribed string (*T*) as the *destination* in Shannon's model. The human-and-text-entry-method serves as the overall transmission channel in Figure 2. If *T* is different from *P*, the message is considered to be perturbed by noise, which is the "error" in the channel. *Throughput* is thus the information transmission rate of the system.

4.1 A Discrete Channel with Noise

In the text entry transcription process, the human-and-text-entry-method transmits a message through a noisy discrete channel: characters are discrete, and errors appear because of noise. Shannon [19] (pp. 407-410) gave a concrete example of the transmission rate of such a channel. In his example, there are two possible symbols, 0 and 1, and they each are produced by the information source with probabilities p_0=p_1= 0.5. The information source transmits the symbols at a rate of 1000 symbols per second. During transmission, noise introduces errors. On average, say, 1 in 100 symbols is received incorrectly, *e.g.*, a 0 is received as a 1, or a 1 is received as a 0. What is the information transmission rate of the channel?

A straightforward answer might seemingly be 990 bits/s, representing the transmission rate of correct symbols. However, because the receiver has no knowledge about *where* the errors occur, this is not the correct answer. (If the receiver knew the location of errors, it would know *everything* about the source, which would mean it has the same information as the source. But this is not the case.)

Transmission rate is the transmitted information per unit time. Suppose X refers to the source and Y refers to the receiver. Shannon defines the "mutual information" $I(X,Y)$ as the transmitted information:

$$I(X,Y) = H(X) - H_Y(X) \quad (2)$$

$H(X)$ represents the information, or "entropy," of the source, and $H_Y(X)$ represents the conditional entropy, which is called "equivocation." It refers to the information of X that could not be gained from Y, the information lost during transmission. More detail can be found in Shannon's original paper [19].

The following equations define entropy and equivocation, respectively:

$$H(X) = -\sum_i p(i) \log_2 p(i) \quad (3)$$

$$H_Y(X) = -\sum_{i,j} p(i,j) \log_2 p_j(i) \quad (4)$$

In Eqs. 3 and 4, i is each symbol in source X, and j is each symbol in receiver Y. The term $p(i)$ is the probability of symbol i being produced by the information source. The term $p(i,j)$ is the probability that the source produces symbol i and symbol j is concurrently received. In the following text, we use i to represent the character produced by the source, and j to represent the character received by the receiver. Moreover, we use $p_a(b)$ to represent the probability of event b *given* that the event a happened. Thus, the term $p_j(i)$ is the probability that the source produces symbol i given symbol j is received, and $p_i(j)$ is the probability that symbol j is received given symbol i is produced.

In Shannon's example, then, $H(X) = -(0.5\ log_2 0.5 + 0.5\ log_2 0.5) = 1.00$ bit per symbol. Furthermore, $p(0,0) = p(1,1) = 0.495$, $p(0,1) = p(1,0) = 0.005$. So $H_Y(X) = -2 \times (0.495\ log_2 0.99 + 0.005\ log_2 0.01) = 0.081$ bits per symbol. Thus, $I(X,Y) = 0.919$ bits per symbol, and the transmission rate is therefore 919 bits/s.

The transmission rate in a text entry transcription process could be calculated as in the example above, if, for now, we were to assume that there were only *substitution errors* [31] in the text entry process (Figure 4). However, in text entry transcription tasks, characters might not only be substituted, but also omitted or inserted. For example, when attempting to type "*abc*", one might type "*ac*" or "*abxc*". How should we deal with *omission* and *insertion* errors [31], which do not have a direct analogue in Shannon's model? We therefore must extend the model.

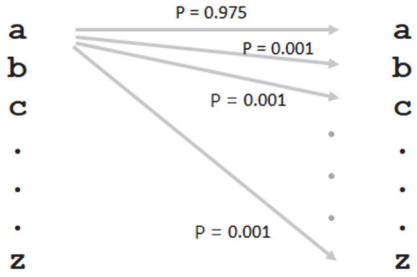

Figure 4. A character transmission probability graph with only *substitution* errors shown. The presented letter is on the left and the transcribed letter is on the right, with given probabilities.

4.2 The Null Character (Ø)

The challenge of handling *omission* and *insertion* errors from an information-theoretic point of view is the lack of a character on either the sending or receiving side of the channel. For omissions, characters are sent by the source but never received. For insertions, characters are never sent by the source but somehow received.

To address this challenge, we extend Shannon's substitution-only model with a *null character*. (We use "Ø" as the notation for the null character in the remainder of this paper.) Using the null character, omission errors occur when a sent character becomes Ø, and insertion errors are when Ø becomes a received character (Figure 5).

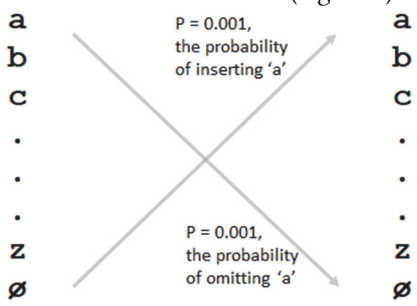

Figure 5. A character transmission probability graph with the null character "Ø" for omission and insertion errors.

Using this approach, we can calculate $p(\emptyset)$, $p(i,\emptyset)$, and $p(\emptyset,j)$ by counting the corresponding *omission* and *insertion* errors, just as for other symbols within the Shannon model. (Note that other related terms, like $p_i(\emptyset)$, can also be calculated using these three terms.)

5 CALCULATION OF THROUGHPUT

Having established how the Shannon model of information transmission maps to text entry transcription tasks—and to substitution, omission, and insertion errors—we are now able to offer detailed steps for calculating *text entry throughput*:

Step 1. Calculate the source information. To calculate the source information $H(X)$, one must get the character distribution of the source according to Eq. 3. In text entry transcription tasks, that means the character distribution $p(i)$ of the presented strings, which could be known by counting the occurrences of each presented character. However, this approach requires large phrase sets to give representative estimates for each character. Furthermore, with the null character in the calculation, there will be a $p(\emptyset)$ in the source distribution. As null is just an imaginary character, including $p(\emptyset)$ would actually lead to different $H(X)$ based on the observed number of omissions and insertions made. The null character should not change the source information, as it is used only for the calculation of the transmission probability.

To address this challenge of computing $H(X)$, we offer a practical solution: fixing the source information on a larger scale—*the language level*. For the English language, we use character-level information because throughput is a character-level information metric, one based on *symbols*[1] in Shannon's original formulation. We use existing probabilities for 26 lowercase letters ('*a*'-'*z*') plus SPACE.[2] We normalize probabilities such that $\sum_{i=1}^{27} p(i) = 1.00$. (The exact probabilities for each letter are in our Appendix.)

Step 2. Calculate transmission probabilities. To get the equivocation $H_Y(X)$, we need to get $p(i,j)$ and $p_j(i)$ for each character i in the information source (presented strings *P*), and each character j in the destination (transcribed strings *T*) according to Eq. 4. Furthermore, we need to accommodate the probability of Ø in this step as well. The new probability distribution of the 27 characters $p'(i)$ that accommodates the null character is $p'(i) = p(i)\big(1 - p(\emptyset)\big)$. The term $p(\emptyset)$ reflects the number of insertion errors over all transmission instances:

$$p(\emptyset) = p'(\emptyset) = \frac{\sum_j N(\emptyset \to j)}{\sum_{i,j} N(i \to j)} \quad (5)$$

[1] We consider English characters as the symbols in this study. Symbols could also be other language units as well, *e.g.*, words or syllables, or language units from non-English languages, such as Chinese logograms.

2 http://www.macfreek.nl/memory/Letter_Distribution

Here, $N(a)$ refers to the number, or count, of event *a*. $N(a \to b)$ refers to the count of character *a* becoming *b*. If we have the transmission probability of character i turning into character j, *i.e.*, $p_i(j)$, then both terms can be calculated as follows. Note that Eq. 7 is an application of Bayes' theorem:

$$p(i,j) = p'(i) \times p_i(j) \tag{6}$$

$$p_j(i) = \frac{p(i,j)}{\sum_i p(i,j)} \tag{7}$$

However, it is not feasible to assume that every possible character i will be substituted by every other possible character j in a laboratory text entry evaluation, given various character frequencies. For example, one might have to type hundreds of phrases to observe one occasion of 'z' being substituted for 'a'. First, some characters like 'z' make few appearances in most phrase sets, leading to few transmission instances of the character. Second, the specific attributes of the method under investigation (for example, the keyboard layout) might lead to few or no instances of substituting character j for character i. These improbabilities mean we cannot measure equivocation precisely unless we make the participant enter myriad phrases.

However, if we move a level above individual characters, the *overall* omission, insertion, and substitution error rate can be observed without too many phrases. Thus, for practical purposes, instead of seeking each $p(i,j)$, we compute the three overall rates, and treat them as equal for all characters as an approximation. The calculation of overall probabilities is shown below:

Overall insertion probability $p(I)$:

$$p(I) = \frac{\sum_{j \neq \emptyset} N(\emptyset \to j)}{\sum_{i,j} N(i \to j)} \tag{8}$$

Overall omission probability $p(M)$:

$$p(M) = \frac{\sum_{i \neq \emptyset} N(i \to \emptyset)}{\sum_{i \neq \emptyset, j} N(i \to j)} \times (1 - p(I)) \tag{9}$$

Overall substitution probability $p(S)$:

$$p(S) = \frac{\sum_{i \neq \emptyset, j \neq \emptyset, i \neq j} N(i \to j)}{\sum_{i \neq \emptyset, j} N(i \to j)} \times (1 - p(I)) \tag{10}$$

Overall probability of correct entries $p(C)$

$$p(C) = \frac{\sum_{i \neq \emptyset} N(i \to i)}{\sum_{i \neq \emptyset, j} N(i \to j)} \times (1 - p(I)) \tag{11}$$

An insertion error happens only when $i = \emptyset$ and is the only error possible when $i = \emptyset$. Thus, there is a $(1 - p(I))$ factor in Eqs. 9-11, as they only can happen when $i \neq \emptyset$.

To get $N(i \to j)$, *i.e.*, how many times presented character i is transcribed as character j, we use work by MacKenzie & Soukoreff [14] to get the optimal alignments between a presented string P and a transcribed string T. When aligning P and T to calculate the minimum string distance (*MSD*), multiple optimal alignments can exist. For example, there are two optimal alignments for P = "optimal" and T = "optiacl", each reflecting *MSD* = 2:

P1: optimal
T1: optiacl

P2: optima-l
T2: opti-acl

In the two alignment pairs above, substitution errors occur at character mismatches; omission errors occur where T has a '−'; and insertion errors occur where P has a '−'. By weighting these errors by the number of times they occur, we can calculate the substitution, omission, insertion and correct-entry probabilities. In the two pairs above, there are two substitutions, one insertion, one omission and eleven correct entries. As there are two alignments in total, the overall probability for substitutions is $[N('m' \to 'a') + N('a' \to 'c')]/2 = 1.0$; for insertions it is $[N(\emptyset \to 'c')]/2 = 0.5$; for omissions it is $[N('m' \to \emptyset)]/2 = 0.5$; for correct entries it is $11/2 = 5.5$.

Step 3. Calculate throughput. Once we have the overall error probabilities for our three error types and correct entries, we get the average probability for each character:

$$p_i(j) = \begin{cases} \frac{p(I)}{N(j)} & i = \emptyset, j \neq \emptyset \\ \frac{p(M)}{N(j)} = p(M) & i \neq \emptyset, j = \emptyset \\ \frac{p(S)}{N(j)} & (i \neq j) \neq \emptyset \\ \frac{p(C)}{N(j)} = p(C) & (i = j) \neq \emptyset \end{cases} \tag{12}$$

In Eq. 12, $N(j)$ is the number of different characters that can be received. For any non-null character i, its probability of omission for each j equals the overall omission probability, because $j = \emptyset$, so $N(j) = 1$. Its probability of correct-entry also equals the overall correct-entry probability, because $j = i$, so $N(j) = 1$. In our experiment, there are 27 symbols ('*a*'-'*z*' and SPACE). Thus, for any non-null character i, its probability of substitution for each j is $p(S)/26$, because there are 26 characters that are different from i. Finally, for $\emptyset$, the probability of insertion of each j is $p(I)/27$, because there are 27 non-null characters that can be inserted.

With Eqs. 6-7, all components for computing the mutual information $I(X,Y)$ can be had according to Eqs. 2-4, which is in units of bits per character. Multiplying $I(X,Y)$ by entry speed, such as characters per second, gets text entry throughput, whose units is in bits per second (bits/s).[3]

In review, to calculate text entry throughput: (1) use general English letter frequencies as the source information; (2) use Ø to handle omission and insertion errors, and align P and T to get the overall probabilities of each type of error; and (3) average the probabilities for each character in four categories: omissions, insertions, substitutions, and correct-entries. Then calculate throughput.

5.1 A Worked Example

To illustrate the calculation process, assume a person transcribes two phrases, P_1 and P_2, as T_1 and T_2, respectively:

```
P1: my watch fell in the water
T1: my wacch fell in water

P2: prevailing wind from the east
T2: previling wind on the east
```

For considerations of space, assume there is only one optimal alignment for each phrase pair:[4]

```
P1: my watch fell in the water
T1: my wacch fell in ----water

P2: prevailing wind from the east
T2: prev-iling wind --on the east
```

In total, there are 7 omission errors, 0 insertion errors, 2 substitution errors, and 46 correct entries. Thus $p(I)$ is 0.000, $p(M)$ is 0.127, $p(S)$ is 0.036, and $p(C)$ is 0.836. The value for $p(\emptyset)$ is 0.000 because there are no insertions.

The source information $H(X)$ is calculated according to English letter frequencies [5] as 4.09 bits/character. After adding the null character Ø, we calculate $p'(i) = p(i)\big(1 - p(\emptyset)\big)$; in this example, $p'(i) = p(i)$, as there are no insertion errors.

According to Eq. 12, for each $i \neq \emptyset$, the substitution probability of i becoming j is $p_i(j) = p(S)/N(j) = 0.036/26 = 0.0014$.

Omission and correct-entry probabilities are the same as their overall respective probabilities. Insertion probability is $0/27 = 0$.

Thus, we can calculate $p(i,j) = p'(i) \times p_i(j)$ for each pair of characters. And $p_j(i) = \frac{p(i,j)}{\sum_i p(i,j)}$. We thus calculate $H_Y(X)$ according to Eq. 4, which yields 0.852. The mutual information is $I(X,Y) = H(X) - H_Y(X) = 3.238$ bits/character. If the entry speed is four characters per second, throughput is $3.238 \times 4.000 = 12.952$ bits/s.

6 STUDY METHOD

We conducted an experiment to put our throughput calculation through its paces using two text entry methods: a laptop keyboard and a smartphone keyboard. We focused on three questions:

1. Can we successfully manipulate different speed-accuracy biases for participants during text entry?
2. How stable is our new throughput metric across different participant speed-accuracy biases?
3. How does throughput stability compare to that of established speed and error rate metrics?

6.1 Participants

Our study was a between-subjects study with a total of 27 participants. Of those, 15 participants (ages 22 – 27, 6 male, 10 female) used a laptop keyboard, and 12 other participants (ages 22 – 25, 6 male, 6 female) used a smartphone keyboard. Recruiting was conducted via email and word-of-mouth. One participant was left-handed; all others were right-handed. All participants indicated years of experience typing with laptop and smartphone keyboards. For the laptop keyboard, participants were compensated $15 USD plus a bonus for about 1.5 hours of their time. For the smartphone keyboard, participants were compensated $15 USD plus a bonus for about 1 hour of their time.

6.2 Apparatus

We compared two input methods: a laptop keyboard and a smartphone keyboard. The laptop keyboard was a Microsoft Surface Pro 4 typecover[6] measuring 11.60" × 8.54" × 0.20". The smartphone keyboard ran on a Google Pixel measuring 2.74" × 5.66" × 0.33". Its keyboard was *Gboard*, a smartphone keyboard with advanced features including opt-out auto-correction, word completion, and a word prediction list. In the study, all advanced features were turned on for the *Advanced* condition, and turned off for the *Plain* condition. Gboard's size measured 2.74" × 2.60".

We ran our study with our new *TextTest++* tool (see Figure 6). *TextTest++* is a web-based text entry method-independent evaluation tool inspired by the popular Windows *TextTest* tool [31]. Timing for each phrase was from the first entered character to the last [11].

3 We offer our code for calculating text entry throughput at https://github.com/DrustZ/Throughput

4 There are actually two optimal alignments for (P_1, T_1) due to the SPACE on either side of presented word "the". There is indeed only one optimal alignment for (P_2, T_2).

5 http://www.macfreek.nl/memory/Letter_Distribution

6 https://www.microsoft.com/en-us/surface/accessories/surface-pro-signature-type-cover

6.3 Procedure

There were five "cognitive sets" [4] that we manipulated in the laptop experiment: *Extremely Accurate (EA)*, *Accurate (A)*, *Neutral (N)*, *Fast (F)*, and *Extremely Fast (EF)*. Each cognitive set represented a different speed-accuracy bias. For example, in *EA*, participants needed to type very carefully to avoid errors, while in *EF*, participants needed to type very fast, even if they might make many errors.

Prior research on the speed-accuracy tradeoff [5] shows that it is not enough to use verbal instructions to impose different cognitive sets. Instead, we designed a game-like text entry task. After participants transcribed a phrase, they received points based on their speed and accuracy relative to the stated objective for that phrase. More points meant a greater cash bonus at the study's end.

Figure 6a shows a text entry transcription trial with a presented string and the transcribed string below it. A 5-minute timer appeared on the left after 10 warm-up phrases were completed. When participants finished entering a phrase, if the total score increased according to the scoring criteria, a green indicator appeared to the right of the text box, accompanied by a "cha-ching" sound; if the score did not change, a black indicator appeared with a "flat" sound; if the score decreased, a red indicator appeared with a "losing" sound. The indicators are shown in Figure 6b.

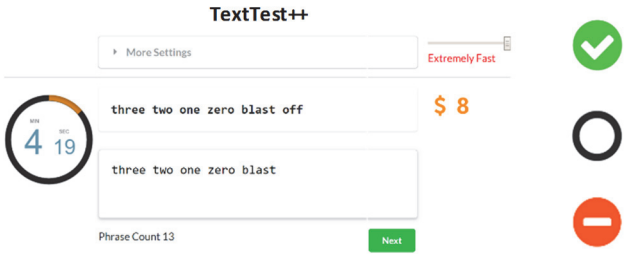

Figure 6. (a) The *TextTest++* interface. The timer is on the left; the condition selector and total score are on the right. After typing in the middle text area, participants could hit the ENTER key or "Next" button to go to the next phrase. (b) Indicators shown after finishing each phrase corresponding to the score increasing, no change, or decreasing, respectively.

Laptop Keyboard. In the laptop keyboard condition, each participant went through two identical blocks of trials. For each block, participants completed transcription trials under the five cognitive sets. For each cognitive set, participants had five minutes to transcribe as many phrases as time allowed. We did not fully counterbalance the order of cognitive sets, as a cognitive set is a relative concept, and thus it was more intuitive for participants to conduct *EA* and *A* adjacently, and likewise *EF* and *F*. We also set *N* to be the last condition, as it was hard to type "neutrally" without having done the accurate and fast conditions first. There were in total 2 (order within *accuracy-biased* group, *i.e.*, *EA-A, A-EA*) × 2 (order within *speed-biased* group, *i.e.*, *EF-F, F-EF*) × 2 (order between *accuracy-* and *speed-biased* groups) = 8 orders in all.

To help participants understand our experiment design, they were taught the scoring mechanism (about which more detail will be given in the next section) before the test, accompanied by verbal instructions (Table 2). Before the study started, participants tried five phrases for each speed-accuracy condition, and tried a 5-minute test in the neutral condition ("*N*") to get familiar with the *TextTest++* user interface and the reward mechanism.

To reduce learning, the phrases in each speed-accuracy condition were different. We mixed the phrase sets from MacKenzie & Soukoreff [15] and Vertanen & Kristensson [27]. We included all 500 phrases of the former source, and extracted 362 phrases without numbers and acronyms from the latter source. The average number of words per phrase was 5.78, and average number of characters per word was 4.97. After participants finished one speed-accuracy condition, the typed phrases were eliminated from subsequent conditions. After each block of trials covering all five speed-accuracy conditions, participants had a rest for five minutes, and then began the second block of trials.

Table 2. Verbal instructions given to participants used in different speed-accuracy conditions.

	Verbal Instruction
EA	"Try to type very accurately and ensure that each key-press is correct."
A	"Try to type accurately. You can make corrections as long as the final string is correct."
N	"Try to type fast and accurately, as in your daily life. A few errors are acceptable."
F	"Try to type fast, and it's OK to make errors."
EF	"Try to type as physically fast as you can, still typing according to the text, but you can ignore errors."

Smartphone Keyboard. In the smartphone keyboard condition, there were two different blocks: one with the advanced features of auto-correction, word completion, and word prediction enabled (*Advanced*), and one with these advanced features disabled (*Plain*). For each block, there were three cognitive sets: *Accurate*, *Neutral*, and *Fast*. We still set *Neutral* as the last condition, resulting in 2 (order between *accuracy-biased* and *speed-biased* conditions, *i.e.*, *A-F, F-A*) × 2 (order between two sessions, *i.e.*, *Advanced-Plain, Plain-Advanced*) = 4 orders in all.

The basic procedure and phrase set were the same as in the laptop keyboard condition. Participants were told to hold the phone and type using a two-handed posture with

two thumbs. We modified *TextTest++* to display properly on a smartphone screen.

6.4 Bonus Mechanism

Figure 7, below, depicts the process for determining participants' bonuses:

Figure 7. Procedure for calculating bonuses for each session.

The points awarded for each phrase in different speed-accuracy conditions are shown in Table 3. Participants received points for a transcribed phrase if they met the accuracy criterion in "Baseline Points." Failing that, they received no points for a phrase if they met the criterion in "No Points." Failing that, they actually *lost* points according to "Lose Points."

Table 3. Points awarded for each phrase in five cognitive sets based on accuracy criterion. The first item in each cell is the points, followed by the criterion to gain or lose those points. *E* refers to the number of uncorrected errors.

	Baseline Points	No Points	Lose Points
EA	+10; $E = 0$ and no corrections	$E = 0$ but corrections > 0	-5; $E > 0$
A	+9; $E = 0$	$0 < E < 5$	-4; $E \geq 5$
N	+8; $E < 5$	$5 \leq E < 10$	-3; $E \geq 10$
F	+7; $E < 10$	$10 \leq E < 15$	-2; $E \geq 15$
EF	+6; $E < 15$	$E \geq 15$	—

In Table 3, *E* refers to the number of uncorrected errors, *i.e.*, the minimum string distance (*MSD*) [24] between the presented string *P* and the transcribed string *T*. As each phrase consisted of a similar number of characters, we measured the error in characters rather than error percentage. Note that for the *EA* condition, along with leaving no errors, participants had to make no error *corrections* in the typing process to gain the baseline points.

As another example, consider the *Accurate* condition (*A*). If there are no uncorrected errors in the transcribed string, the participant will get 9 points. In the *Neutral* condition (*N*), a participant would get zero points if there are 5 errors in the transcribed string, and would lose 3 points if there are 10 or more uncorrected errors.

The allocated testing time for each condition was fixed; thus, in order to get as high a score as possible, participants had to type as many phrases as possible, while still meeting the accuracy criterion for points. This scheme ensured that participants would strive for their best performance in each condition, rather than type unnecessarily slowly to meet the accuracy criterion for each phrase.

The bonus table for the laptop keyboard is given in Table 4, which shows expected minimum speeds (WPM_e) and the total required score (S_r; see Eq. 13) for each condition. (For the smartphone keyboard, we halved the laptop speeds in the corresponding *A/N/F* conditions.)

Table 4. Bonuses based on the expected speeds (WPM_e) and required scores (S_r) (Eq. 13) for the five speed-accuracy conditions. Each cell contains two values, the minimum expected typing speed (WPM_e) and the required score (S_r) to get the corresponding bonus. WPM_e is used to calculate S_r.

	WPM_e | S_r thresholds for given bonuses				
EA	15 | 125	35 | 425	65 | 875	105 | 1475	145 | 2075
A	20 | 170	40 | 440	70 | 845	110 | 1385	150 | 1925
N	25 | 200	45 | 440	75 | 800	115 | 1280	155 | 1760
F	30 | 215	50 | 425	80 | 740	120 | 1160	160 | 1580
EF	35 | 215	55 | 395	85 | 665	125 | 1025	165 | 1385
$$$	**$1**	**$2**	**$3**	**$4**	**$5**

For each speed-accuracy condition, a participant's total points are summed up, and compared to the calculation of the required score (S_r) for that condition, as follows:

$$S_r = \frac{WPM_e \times TM \times BP}{AWP} \times 0.75 - 100 \qquad (13)$$

In Eq. 13, *TM* is total minutes, *BP* is baseline points from Table 3 (thus ranging from +6 – +10), and *AWP* is average words per phrase. Based on our phrase set and experiment procedure, we set *TM* to 10 and *AWP* to 5. The reason for the term "× 0.75 – 100" is based on our pilot study to make it easier to get bonuses.

In general, then, participants received an $\$N$ reward if their total points was above the required score (S_r) shown in Table 4. For example, the score in the *Neutral* (*N*) condition for a reward of \$3 is 800; for \$4, it is 1280. Thus, if one reached 895 points in condition *N*, one would get a \$3 bonus for that condition.

Although the payoff scheme might seem a bit complicated, it was straightforward for participants: from *EA* to *EF*, they got fewer points for each phrase, but also a smaller error penalty with more room to make errors. We ensured all participants understood *EA-EF* before the experiment, having them demonstrate their typing strategy in each condition.

Each participant was supposed to get at least a \$1 bonus for each speed-accuracy condition; thus, we made the corresponding typing speeds slow. In contrast, we made the speeds for a \$5 bonus high so that this level would not be exceeded by even an expert typist. As a result, all participants had reason to try hard to gain points.

7 RESULTS

We compared four metrics: speed (WPM) [11], uncorrected error rate [25], adjusted speed (AdjWPM) [30], and throughput in both keyboard conditions. Throughput exhibited the least variance across different speed-accuracy conditions, and displayed a mild bell-shaped pattern from *Extremely Accurate (EA)* to *Extremely Fast (EF)*. Prior studies of throughput in Fitts' law also show throughput to be reasonably stable across different speed-accuracy biases, but never invariant (see, *e.g.*, [4,9,12,33]).

7.1 Laptop Keyboard Condition

In all, 7342 phrases were collected in the laptop keyboard condition. There were 2101 phrases from warm-up and practice sessions, which were not included in our analysis. The average uncorrected error rate, speed, AdjWPM (see Eq. 1), and throughput of the 16 participants are shown in Figure 8.

As Figure 8 shows, our reward mechanism worked well on manipulating different cognitive sets. From *EA* to *EF*, participants left more errors and typed faster. Errors and speed varied a lot from *EA* to *EF*, but throughput was quite stable across conditions.

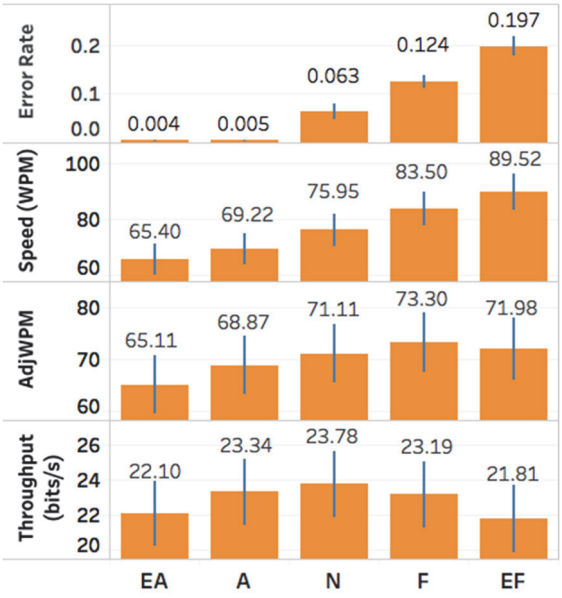

Figure 8. Results for the laptop keyboard in five cognitive sets, from *Extremely Accurate* to *Extremely Fast*. Error bars represent ±1 standard error. Note the very different *y*-axis ranges, which are set for visual comparison of differences within each metric across speed-accuracy conditions.

We used *coefficient of variation (CV)* to measure the variance of groups of data that have different units. *CV* equals the ratio of the standard deviation to the mean: the smaller the *CV*, the less varied the data. We calculated the *CV* of the four metrics as: 1.05 for error rate; 0.13 for speed, 0.04 for AdjWPM and 0.03 for throughput. Thus, AdjWPM varied less than speed (raw WPM) but still more than throughput. Interestingly, the bell-shaped curve indicates that throughput was highest in the *Neutral* condition, and dropped when it was heavily accuracy- or speed-biased. Not so for AdjWPM, which generally increased with WPM, indicating that it was not compensating much for increasing errors, as throughput was. The shape of throughput results across speed-accuracy conditions aligns with the findings of Liu *et al.* [9] on command selection, which showed a similar pattern for throughput in pointing.

To quantitatively evaluate our results, we performed a non-parametric analysis on the four text entry metrics. Friedman tests showed there was a significant main effect of speed-accuracy condition on error rate ($\chi^2_{(4,N=80)} = 56.65$, $p < .001$), speed ($\chi^2_{(4,N=80)} = 61.00$, $p < .001$), AdjWPM ($\chi^2_{(4,N=80)} = 23.15$, $p < .001$), and throughput ($\chi^2_{(4,N=80)} = 23.25$, $p < .001$). For error rate and speed, Wilcoxon signed-rank tests using Holm's sequential Bonferroni procedure [6] to correct for multiple comparisons showed that every condition was significantly different from every other ($p < .005$). However, for AdjWPM, there were no significant differences between the *Neutral* and *Fast* or *Neutral* and *Accurate* conditions, but *Fast* and *Accurate*, and *Extremely Fast* and *Extremely Accurate*, were significantly different ($p < .05$). As shown in Figure 8, AdjWPM showed an ascending trend from *EA* to *F*, with faster conditions generating higher AdjWPM.

The significant effect of speed-accuracy condition on throughput also led us to investigate pairwise differences. Using Wilcoxon signed-rank tests corrected with Holm's sequential Bonferroni procedure [6], we found no significant differences between each pair of *Accurate*, *Neutral*, and *Fast*. Interestingly, there was also no significant difference between *Extremely Accurate* and *Extremely Fast*, indicating that throughput decreased similarly in both extreme conditions. The two *Extreme* conditions were significantly different from the *Neutral* condition ($p < .05$), which was reasonable because there were unnatural constraints in the *Extreme* conditions: in the *EA* condition, one had to press each key correctly to get points; in the *EF* condition, one had to type as physically fast as one could. Similarly, there was a significant difference between *EA* and *A* ($p < .05$), and *EF* and *F* ($p < .05$).

To provide further evidence of the stability of throughput across speed-accuracy conditions, we applied bootstrapping on the experiment data. Bootstrapping is a statistical test that randomly samples the data with replacement. It is useful for small data sets, and can estimate confidence intervals (*CI*). To test how similar throughput is between two conditions, we subtract one condition from

another, resulting in a new data set of differences. We then resampled the data 10,000 times and derived an estimated 99.5% *CI* from 0.25% to 99.75%. The interval was corrected with the Holm-Bonferroni correction [6], as there were 10 pairwise comparisons among the speed-accuracy conditions in total. If zero was located inside the *CI*, there might not be a significant difference between any two conditions. The results from bootstrapping were similar to the significance testing results. No significant differences were found between the *A/N/F* conditions, or between the *EA/EF* conditions (*N-A 99.5% CI* [-0.18, +1.04]; *N-F 99.5% CI* [-0.31, +1.88]; *A-F 99.5% CI* [-1.00, +1.43]; *EA-EF 99.5% CI* [-1.24, +2.01]). Collectively, these results indicate that our throughput measure is quite stable across speed-accuracy conditions.

7.2 Smartphone Keyboard Condition

In all, 4811 phrases were collected on the smartphone. There were 1494 phrases from warm-up and practice sessions, which were not included in our analysis. The average uncorrected error rate, speed, AdjWPM, and throughput are shown in Figure 9.

Non-parametric Friedman tests showed that there were significant differences in throughput among different speed-accuracy conditions for both the *Advanced* ($\chi^2_{(2,N=36)} = 12.50$, $p < .005$) and *Plain* ($\chi^2_{(2,N=36)} = 17.17$, $p < .001$) keyboards. This led us to perform *post hoc* pairwise comparisons using Wilcoxon signed-rank tests corrected with Holm's sequential Bonferroni procedure [6]. We found that with the *Advanced* keyboard, there was a significant difference between *N/A* ($p < .05$), but no significant difference between *A/F* and *N/F*. With the *Plain* keyboard, the difference was significant between *N/A* and *A/F* ($p < .05$), but not between *N/F*. For other metrics, pairwise comparisons between the *A/N/F* conditions were significantly different in error rate and speed for both smartphone keyboards. For AdjWPM, the only non-significantly different pairwise comparison was *N/F* for the *Advanced* keyboard.

Thus, although throughput showed an ascending trend with entry speed, the difference between each speed-accuracy condition was smaller than for the other text entry metrics.

Interestingly, Wilcoxon signed-rank tests also showed that there was no significant difference between the two keyboards in any cognitive set condition, which means that performance was not significantly different whether typing with or without auto-correction, word completion, and word prediction.

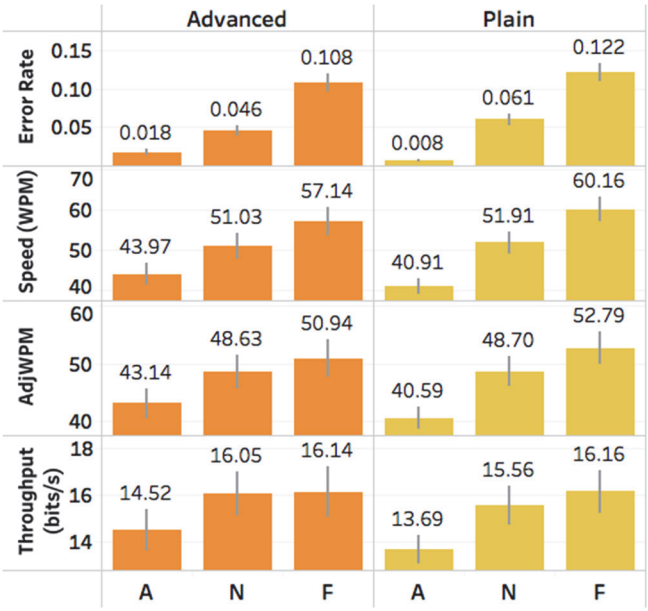

Figure 9. Results for the smartphone keyboard in three cognitive sets, from *Accurate* to *Fast*. The *Advanced* plots are for the keyboard with auto-correction, word completion, and word prediction. The *Plain* plots are for the same keyboard without these advanced features. Error bars represent ±1 standard error. Note the very different *y*-axis ranges, which are set for visual comparison of differences within each metric across speed-accuracy conditions.

8 DISCUSSION

Our laptop keyboard condition provided empirical support for the practical value of our new throughput metric and its estimation for text entry. From the consistent results under neutral, speed-biased, and accuracy-biased conditions, we showed that throughput was relatively stable. Because the *EA/EF* conditions were intentionally extreme and unnatural, it was reasonable that throughput decreased in such conditions. Actual text entry evaluations will not generally require participants to perform in such unnatural ways, so the practical impact of the *EA* and *EF* conditions is minimal.

For both keyboards, the *F/EF* conditions had higher AdjWPM than in other conditions, indicating that the faster participants typed, the better they performed under this metric, even though their error rates increased significantly. This finding is evidence that AdjWPM is speed-biased and fails to adequately correct for increasing errors. In contrast, our new throughput metric showed a bell-shaped performance curve in the laptop keyboard condition, indicating the existence of a speed-accuracy tradeoff. As the difficulty of the task increased, participants no longer typed within their comfortable performance range, and thus their throughput decreased. The same conclusion was also

reached in prior work [9,22], albeit for tasks other than text entry.

In the smartphone keyboard condition, although the *Accurate* condition had the lowest throughput, that was reasonable considering that making corrections and typing every character correctly on a touch screen was more difficult than doing so on a physical keyboard. It was also interesting that there was no significant increase in throughput using auto-correction, word completion, and word prediction. Looking at the average speed and error rate, we could even see drops in performance in some conditions with these advanced features. Some participants reported that they felt distracted looking at and selecting from a word list during typing, which might have increased their entry time. One participant reported that each time after she pressed a key, she spent time searching the word list to see if there was an expected word. Thus, the uncertainty brought about by the advanced features might have increased the response time of participants. Similar findings about how advanced features can harm typing performance have been reported elsewhere [7].

On the whole, our experiment evaluated our new throughput metric for text entry across different speed-accuracy conditions using a differential reward mechanism. All of our participants understood the bonus scheme and how to maximize their points in each condition. However, they found it hard to distinguish between *Fast* and *Extremely Fast* conditions, which indicated that it was generally difficult to manipulate more than three cognitive sets (*Accurate/Neutral/Fast*) unless there was a clear difference between the criteria in each condition. From our observations, most participants typed in an accuracy-biased manner—they tended to correct most errors. When typing in a speed-biased manner, they first felt uncomfortable letting errors remain.

As a metric, throughput also exhibited other desirable properties: the data-collection process was exactly the same as in current text entry evaluation processes [13], and the calculation was straightforward following the steps we outlined above.

8.1 Limitations

In our throughput algorithm, we average the overall probability of *insertion, omission, substitution* and *correct-entries* to calculate each $p(i \rightarrow j)$, as the throughput metric is method-independent. However, researchers dealing with their own specific text entry methods might want to get the exact probability of each error instance. For example, one could simulate the transmission probability based on the interface (*e.g.*, keyboard layout). Using overall error rates as an approximation for each character-level error is the major limitation of this work—but as limitations go, it is a practical, not theoretical, one.

9 FUTURE WORK

We see at least four possible directions for future work. First, one could experiment with different input methods, which might include speech, gesture, and other new text entry interfaces. In this study, we only evaluated keyboards. However, our throughput metric will work with any text entry method. Second, one could experiment with other languages in which the character information and the input method are different from English. For example, Chinese is a logogram-based language very different than a phonogram-based language like English. Third, in the throughput calculation, one could use entropy on higher-level language constructs, such as the word-level. For example, Shannon [20] pointed out that English word-level entropy is 11.82 bits per word. This might be useful for evaluating the "semantic performance" of a method. Fourth and finally, there could be more accurate and robust ways of estimating throughput other than the error category estimation approach we took in this study.

10 CONCLUSION

In this work, we presented a new text entry method-independent unified speed-accuracy metric *throughput* built on Shannon information theory. Our experiment on laptop and smartphone keyboards showed that throughput was stable across different speed-accuracy biases compared to other metrics like error rate, words per minute, and adjusted words per minute. Our hope is that throughput will be calculated and reported to support comparisons across devices, text entry methods, and participants. The advantageous properties of throughput and the theoretical derivation from information theory make it suitable as a performance-level metric. At the same time, speed and accuracy should also be reported, as they provide essential practical insight, even if they are more dependent than throughput on task-specific experimental conditions.

ACKNOWLEDGMENTS

We thank Paul V. Roby for materials on the speed-accuracy operating characteristic. This work was supported in part by Baidu. Any opinions, findings, conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect those of any supporter.

REFERENCES

[1] Ahmed Sabbir Arif and Ali Mazalek. 2016. WebTEM: A Web Application to Record Text Entry Metrics. *Proceedings of ACM ISS 2016*. ACM, New York, 415–420.

[2] E. R. F. W. Crossman. 1957. The Speed and Accuracy of Simple Hand Movements. *The Nature and Acquisition of Industrial Skills*.
[3] Paul M. Fitts. 1954. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology 47*, 381–391.
[4] Paul M. Fitts and Barbara K. Radford. 1966. Information Capacity of Discrete Motor Responses Under Different Cognitive Sets. *Journal of Experimental Psychology 71*, 475–482.
[5] Richard P. Heitz. 2014. The speed-accuracy tradeoff: history, physiology, methodology and behavior. *Frontiers in Neuroscience* 8, Article 150, 1–19.
[6] Sture Holm. 1979. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics* 6, 2: 65–70.
[7] Heidi Horstmann Koester and Simon Levine. 1996. Effect of a word prediction feature on user performance. *Augmentative and Alternative Communication* 12, 3: 155–168.
[8] James R. Lewis. 1999. Input rates and user preference for three small-screen input methods: Standard keyboard, predictive keyboard, and handwriting. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 425–428.
[9] Wanyu Liu, Olivier Rioul, Michel Beaudouin-Lafon, and Yves Guiard. 2017. Information-Theoretic Analysis of Human Performance for Command Selection. *Proceedings of INTERACT 2017*. Lecture Notes in Computer Science vol. 10515, Springer, 515–524.
[10] I. Scott MacKenzie. 1992. Fitts' Law As a Research and Design Tool in Human-computer Interaction. *Human-Computer Interaction* 7, 1: 91–139.
[11] I. Scott MacKenzie. 2002. A Note on Calculating Text Entry Speed. http://www.yorku.ca/mack/RN-TextEntrySpeed.html
[12] I. Scott MacKenzie and Poika Isokoski. 2008. Fitts' Throughput and the Speed-accuracy Tradeoff. *Proceedings of ACM CHI 2008*. ACM, New York, 1633–1636.
[13] 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: 147–198.
[14] I. Scott MacKenzie and R. William Soukoreff. 2002. A Character-level Error Analysis Technique for Evaluating Text Entry Methods. *Proceedings of ACM NordiCHI 2002*. ACM, New York, 243–246.
[15] I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase sets for evaluating text entry techniques. *Extended Abstracts of ACM CHI 2003*. ACM, New York, 754–755.
[16] I. Scott MacKenzie and Shawn X. Zhang. 1999. The design and evaluation of a high-performance soft keyboard. *Proceedings of ACM CHI 1999*. ACM, New York, 25–31.
[17] Tim Paek and Bo-June (Paul) Hsu. 2011. Sampling Representative Phrase Sets for Text Entry Experiments: A Procedure and Public Resource. *Proceedings of ACM CHI 2011*. ACM, New York, 2477–2480.
[18] Richard W. Pew. 1969. The Speed-accuracy Operating Characteristic. *Acta Psychologica* 30: 16–26.
[19] Claude E. Shannon. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal* 27, 3: 379–423.
[20] Claude E. Shannon. 1951. Prediction and entropy of printed English. *The Bell System Technical Journal* 30, 1: 50–64.
[21] R. William Soukoreff. 2010. *Quantifying Text Entry Performance*. Ph.D. Dissertation. York University, Canada.
[22] R. William Soukoreff and I. S. MacKenzie. 2009. An informatic rationale for the speed-accuracy trade-off. *Proceedings of IEEE Conference on Systems, Man, and Cybernetics*. IEEE, New York, 2890–2896.
[23] R. William Soukoreff and I. Scott Mackenzie. 1995. Theoretical upper and lower bounds on typing speed using a stylus and a soft keyboard. *Behaviour & Information Technology* 14, 6: 370–379.
[24] R. William Soukoreff and I. Scott MacKenzie. 2001. Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic. *Extended Abstracts of ACM CHI 2001*. ACM, New York, 319–320.
[25] 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. *Proceedings of ACM CHI 2003*. ACM, New York, 113–120.
[26] Dan Venolia and Forrest Neiberg. 1994. T-cube: A Fast, Self-disclosing Pen-based Alphabet. *Conference Companion to ACM CHI 1994*. ACM, New York, 265–270.
[27] Keith Vertanen and Per Ola Kristensson. 2011. A Versatile Dataset for Text Entry Evaluations Based on Genuine Mobile Emails. *Proceedings of ACM MobileHCI 2011*. ACM, New York, 295–298.
[28] A T Welford. 1968. Movement. In *Fundamentals of Skill*. Methuen, London, England, 137–160.
[29] Jacob O. Wobbrock and Brad A. Myers. 2006. Trackball Text Entry for People with Motor Impairments. *Proceedings of ACM CHI 2006*. ACM, New York, 479–488.
[30] Jacob O. Wobbrock. 2007. Measures of text entry performance. In *Text Entry Systems: Mobility, Accessibility, Universality*, I S MacKenzie and K Tanaka-Ishii (eds.). San Francisco: Morgan Kaufmann, 47–74.
[31] 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: 458–489.
[32] Xin Yi, Chun Yu, Mingrui Zhang, Sida Gao, Ke Sun, and Yuanchun Shi. 2015. ATK: Enabling Ten-Finger Freehand Typing in Air Based on 3D Hand Tracking Data. *Proceedings of ACM UIST 2015*. ACM, New York, 539–548.
[33] Shumin Zhai, Jing Kong, and Xiangshi Ren. 2004. Speed-accuracy Tradeoff in Fitts' Law Tasks: On the Equivalency of Actual and Nominal Pointing Precision. *International Journal of Human-Computer Studies* 61, 6: 823–856.
[34] Shumin Zhai and Per-Ola Kristensson. 2012. The word-gesture keyboard. *Communications of the ACM* 55, 9: 91–101.

APPENDIX

English letter frequencies used in this paper, adapted from http://www.macfreek.nl/memory/Letter_Distribution. We included SPACE and normalized frequencies to sum to 1.000.

Letter *i*	***p(i)***
a	*0. 06545420428810268*
b	*0. 012614349400134882*
c	*0. 022382079660795914*
d	*0. 032895839710101495*
e	*0. 10287480840814522*
f	*0. 019870906945619955*
g	*0. 01628201251975626*
h	*0.0498866519336527*
i	*0.05679944220647908*
j	*0.0009771967640664421*
k	*0.005621008826086285*
l	*0.03324279082953061*
m	*0.020306796250368523*
n	*0.057236004874678816*
o	*0.061720746945911634*
p	*0.015073764715016882*
q	*0.0008384527300266635*
r	*0.049980287430261394*
s	*0.05327793252372975*
t	*0.07532249847431097*
u	*0.022804128240333354*
v	*0.007977317166161044*
w	*0.017073508770571122*
x	*0.0014120607927983009*
y	*0.014305632773116854*
z	*0.0005138874382474097*
SPACE	*0.18325568938199557*
TOTAL	*1.00000*