Design and Analysis of Intelligent Text Entry Systems with Function Structure Models and Envelope Analysis

Per Ola Kristensson
Department of Engineering
University of Cambridge
Cambridge, United Kingdom
pok21@cam.ac.uk

Thomas Müllners
Department of Engineering
University of Cambridge
Cambridge, United Kingdom
tdm33@cam.ac.uk

ABSTRACT

Designing intelligent interactive text entry systems often relies on factors that are difficult to estimate or assess using traditional HCI design and evaluation methods. We introduce a complementary approach by adapting function structure models from engineering design. We extend their use by extracting controllable and uncontrollable parameters from function structure models and visualizing their impact using envelope analysis. Function structure models allow designers to understand a system in terms of its functions and flows between functions and decouple functions from function carriers. Envelope analysis allows the designer to further study how parameters affect variables of interest, for example, accuracy, keystroke savings and other dependent variables. We provide examples of function structure models and illustrate a complete envelope analysis by investigating a parameterized function structure model of predictive text entry. We discuss the implications of this design approach for both text entry system design and for critique of system contributions.

CCS CONCEPTS

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

KEYWORDS

text entry design, predictive text entry, computational interaction, engineering design, computational modeling

ACM Reference Format:
Per Ola Kristensson and Thomas Müllners. 2021. Design and Analysis of Intelligent Text Entry Systems with Function Structure Models and Envelope Analysis. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3411764.3445566

1 INTRODUCTION

It is an incredible challenge to design, analyze and evaluate human-computer interaction (HCI) systems (e.g. [21]) and in particular intelligent text entry systems that have to infer or predict users' actions [17]. One view of design in systems engineering is that design is about choosing an operating point in a multidimensional design space. The trade-offs in choosing a particular operating point are either explicit (the designer is aware of all critical factors) or implicit (the designer is unaware of one or more factors and the trade-offs are decided implicitly rather than explicitly by the designer). The central contribution in this paper is to propose *parameterized function structures* coupled with *envelope analysis* to assist the design of intelligent text entry systems by visualizing the consequences of choosing certain operating points in such a design space.

The approach allows text entry system designers to 1) arrive at a system map of key functions and parameters; and 2) use this map to quantitatively model and visualize effects of key controllable and uncontrollable parameters. This allows system designers to understand the ways the system's emergent behavior is sensitive to parameter choices. Crucially, this approach makes no attempt to explain or predict actual user behavior directly. It is therefore a *complementary* approach to traditional HCI design and evaluation methods. The approach results in an emergent model that generates *all* design futures for the system: the model is therefore only incorrect if the underlying parameterization is fundamentally incorrect.

As a running example of this system design approach, we use predictive text entry. Predictive text keyboards use a statistical language model to predict what the user is typing and presents these as shortcut options to the user. On mobile devices this is intended to not only make typing easier and more reliable, but also to improve the text entry rate.

Whilst in theory the keystroke savings provided by predictions can facilitate faster text entry, in practice the outcome is not always favorable, as attending to and selecting suggestions comes at a cognitive cost. There are many interface and user-dependent factors that influence text entry [8]. Understanding these factors and how they interrelate is critical to making informed decisions for improved system designs. Several studies have observed that there is a trade-off between costs and benefits in using predictions [1, 2, 13, 24]. However, the conditions under which the benefits outweigh the costs are not well understood.

The ability of the language model to make reliable predictions plays a pivotal role in the performance of any predictive text entry method. However, to date no study has systematically controlled for the influence of the language model or related the mechanisms of word prediction back to system design. A research question that motivates this work is a desire to gain a better understanding of the costs and benefits of predictive text entry, the effects of the language

CHI '21, May 8–13, 2021, Yokohama, Japan
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8096-6/21/05.
https://doi.org/10.1145/3411764.3445566

model and how these affect the system design aspects of predictive text. To address this research question we adopt a parameterized function model commonly used in engineering design [22] to model hypothetical benefits of word prediction from a systems perspective. We use this function model to derive a computational model of predictive text entry that allows us to explore the performance envelopes of text prediction.

In this paper we therefore present a computational model of typing with word predictions and use it to carry out such an envelope analysis. We investigate three parameterized fundamental typing strategies: minimum word length, type-then-look and perseverance. Among other things, our analysis allows us to identify critical aspects of the design, such as the importance of the ratio of the physical and cognitive task parameters and the observation that performance gains are only realized when typing at least two letters before looking at predictions. We also discuss how function models can in general be used to guide research and design of AI-infused text entry systems.

The central contributions of this paper are the following:

- Introducing function structure modeling to text entry system design and demonstrating the usefulness of mapping out functions and flows of signals between functions before embarking on translating functions to function carriers.
- Explaining how such a model can be parameterized into a generative model and concretely demonstrating this approach by creating a function model for predictive text systems.
- Demonstrating how this approach can be furthered to generate a quantitative model that allows design parameter exploration to understand possible design futures within a system envelope.
- Discussing the benefits of this approach to 1) avoid common errors in text entry system design; and 2) assist in evaluating system research.

2 RELATED WORK

The limitations of traditional HCI evaluation methods for design have been reflected on eloquently before [11, 19]. In particular, Olsen [21] carefully considered the many issues involved in evaluating systems research. This paper aims to support design and evaluation of intelligent text entry systems design by proposing a *complementary* design approach that leverages techniques adapted from design engineering to text entry system design. Specifically we will adopt a view of design that decouples *functions* (*what* is carried out) from *function carriers* (*how* is it carried out). To assist this view of design we will adapt *function structure models*, a widely used technique in design engineering (e.g. [22]). Function structure models allow text entry system designers to arrive at a schematic system design view of key aspects of a design by identifying one or several key functions and subsequently decomposing them into critical subfunctions. Signal flows between these functions are indicated by arrows (for two examples, see Figure 2). We introduce function structures into HCI by parameterizing them with the most important design parameters. As will be explored later in this paper, this allows system designers to perform quantitative analysis that allow visualization of the emergent *envelope* of a system. Kristensson et al. [18] used such envelope analysis to assess the performance of a sentence retrieval system for nonspeaking individuals with motor disabilities. This paper expands that work by proposing and demonstrating parameterized function structure models and envelope analysis for general text entry system design.

We will use predictive text to demonstrate the efficacy of the approach. As a consequence we review related work in this application area. A recent paper by Quinn and Zhai [24] compared the performance of mobile keyboards with different ways of presenting predictions to the user: *introverted* (no suggestions), *extroverted* (always making suggestions) and *ambiverted* (more reserved in making suggestions). It demonstrated that the cost of demanding the user's attention to attend to suggestions can have a negative impact on text entry rate—the variant that showed suggestions most frequently also achieved the slowest entry rates, despite an increase in keystroke savings. The authors acknowledge that the observed cost-benefit balance of using predictions would likely shift under different conditions. We complement this work by simulating performance envelopes under certain modeling assumptions, including incorporating the *strategy* of the user. A later user study [1] also confirmed that word predictions can have a negative effect on entry rate, although the impact of the language model was not studied and the authors acknowledge that the outcome of the study would likely change if a more advanced word prediction model was used.

Early work in characterizing and modeling predictive text systems, in the context of assistive technology aimed at individuals with motor disabilities, was carried out by Koester et al. in the 1990s [12–15]. These studies observed that the cognitive cost of attending to word predictions often outweighed the benefits of keystroke savings.

In an orthogonal dimension, the physical aspect of typing on touchscreen keyboards has been studied and modeled extensively [3, 4, 26]. However, these studies do not incorporate any of the processes associated with attending to word predictions.

A few papers have looked at modeling performance of text predictions [7, 9, 12]. This prior work essentially presented a phrase set to a language model letter-by-letter and recorded the number of letters needed before a correct prediction was found, which was then used to compute an average achievable keystroke savings across a phrase set. Keystroke savings is a metric which is often quoted in this field and although it gives an indication of how capable the language model is in aiding predictive typing, it is calculated without any consideration of the process by which text is entered in practice. As noted in many studies, the use of predictions comes at a cognitive cost, which may or may not be offset by the time savings it supposedly provides. Without this context, metrics such as keystroke savings or error rate do not provide a good indication of entry rate performance in practice.

There is an awareness, but also a lack of understanding, of the cognitive costs of using predictions and how this cost trades off against the physical process of typing. Much of the experimental research treats text entry as a black box, where many factors, not least of which the language model, are not adequately controlled. The conclusions from these studies could very likely change under different language models [2, 24].

Koester et al. [15] studied five different typing strategies, Liu et al. [20] modeled entry of Chinese pinyin and Silfverberg et al. [25] investigated text entry for phone keypads. These studies only considered a limited scope of typing strategies, and Koester et al. [15] is the only work that makes an attempt at controlling for the influence of the language model.

Koester et al. [15] sets out to produce a model that can be used to predict what the most effective typing strategy is for a given user characteristic (keystroke and cognitive variables) and system characteristic (keystroke savings, as facilitated by the language model). Three word prediction conditions were considered, characterized by different levels of average keystroke savings: *low*, *average* and *high*. Crucially, however, the method for controlling keystroke savings was not by changing the actual language model, but rather by manipulating the phrase set used as input to the simulation, with simpler phrase sets resulting in higher keystroke savings. These three tiers of phrase sets had to be produced in a prior empirical study [12]. For each tier, the input phrase set was adjusted until the results showed the desired level of keystroke savings, again, with simpler phrase sets resulting in higher keystroke savings. The resulting phrase sets are therefore tightly coupled to the system conditions, the typing strategy and the language model under which the experiment was conducted. It is uncertain whether this approach actually produces internally and/or externally valid data, and whether the simulation model can even be generalized. More importantly, however, since the approach to controlling the influence of word predictions relies entirely on empirical studies to be conducted in advance, it severely limits its practical application as a theoretical design and modeling tool. Adding a new operating point to this model would require conducting a new user study in order to obtain the necessary phrase set.

3 APPROACH

We here introduce a design method based on function structure models from engineering design [22] to the design of intelligent text entry systems. This design method is typically used at the conceptual design stage in product development to allow design engineers to better understand the system design in terms of its functions and flows between functions. Importantly it decouples *functions* from *function carriers*. We will use function structures to extract controllable and uncontrollable parameters that govern any function at a purely functional level before committing to particular function carrier solutions.

Figure 1 illustrates the overall process, which may be iterated, adapted to intelligent text entry system design. Requirements and design objects, design know-how, etc. is used to first arrive at a purely functional model of the system (see Figure 2 for two examples of such function structure models). This *functional description* describes the *what*—what needs to be done by the system in order to carry out its overall function, such as supporting a user in typing a word. The functional description is then transformed into a detailed description which describes the *how*—how does the system implement the functions in the functional description. In other words, the process transforms *functions* into *function carriers*—implementations of functions. Once all function carriers have been decided the detailed description is implemented as a system. At this point the system can be verified (checking that all requirements have been met) and validated (evaluating the system in terms of its benefits to end-users).

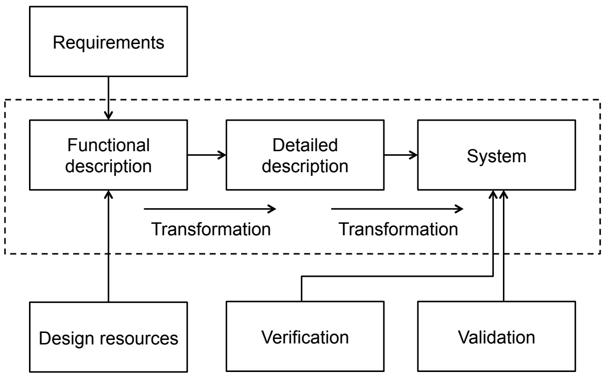


Figure 1: An illustration of the engineering design process adapted for the design of a software-based interactive system, in particular an intelligent text entry system.

In principle, it should be uncontroversial to apply this (simplified) description of an engineering design process to the design of intelligent text entry systems. However, we note two common errors and their consequences.

System design error 1: failing to design a functional description. The first error is to ignore the functional description step and address the detailed description immediately. As we will see in this paper, such a direct approach makes it difficult to carry out an informed analysis of the underpinning *mechanisms* that in the end will govern end-user performance.

System design error 2: carrying out short-circuit evaluations. The second error is to determine the merits of a single design choice, such as whether to supply users with word prediction, by evaluating a particular design choice using a complete system without fully understanding the implications of the fundamental parameter choices in the underpinning function model. We call this error *short-circuit evaluation*. The problem with short-circuit evaluation is that any result is contingent on a large number of factors and fixed underlying design parameters. For example, evaluating the merits of word prediction on a touchscreen phone running a particular built-in system keyboard, means the observed performance may be dependent on the interaction context, the form factor of the phone, the language model, the touch model, the pose and the situation the user is typing in (walking, sitting, standing), how word predictions are displayed, the particular text the user is typing, whether users are transcribing or composing text, etc.

In reality, empirical evaluation is critical for text entry progress and we are not advocating revisiting the HCI debate about the merits of empirical research (e.g. [11, 19]). However, there are limits on how many factors that can be reliably controlled in any user study and it is sometimes difficult to understand how sensitive any obtained results may be to certain underlying fixed parameter choices that underpin a particular text entry system used in an evaluation. This paper strives to demonstrate that parameterized function structure models in conjunction with computational

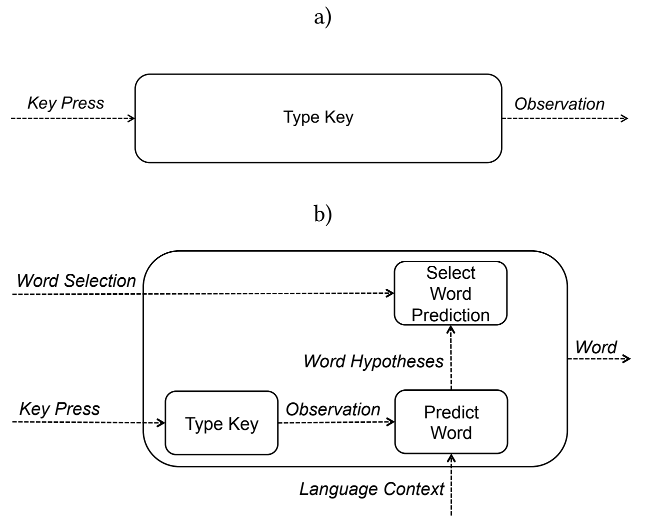


Figure 2: Two function structures models for text entry. Dashed arrows indicate signal flows. (a) The overall system function `Type Key` outputting an *Observation* signal in response to a *Key Press* signal. (b) The overall system function `Type Word` decomposed into three main functions.

models are useful *complementary* system design tools in text entry design to help better understand the underpinning mechanisms before committing to a particular implementation or user study design.

4 FUNCTION STRUCTURE MODELS

A function structure model is a graphical technique for illustrating function and signal dependencies in a design. Figure 2 (a) illustrates the function structure for the overall system function `Type Key`. Dashed arrows reveal input and output signals as well as signal flows in the internal design (visible in the decomposed function structure (b) in Figure 2). Figure 2 (a) shows that `Type Key` outputs an *Observation* signal in response to a *Key Press* signal.

The overall process for arriving at a parameterized function structure model and subsequent analysis consists of six steps.

(1) Identify the overall function, for example, `Type Key` in Figure 2 (a).
(2) Connect any applicable signals. For example, `Type Key` has one *Key Press* input signal and one *Observation* output signal.
(3) Parameterize the function and its signals. For example, *Key Press* can be parameterized as a pair consisting of the corresponding letter of the key pressed and the key press time duration.
(4) Decompose the function into subfunctions and expand the function structure with additional functions, as required. For example, Figure 2 (b) shows a function `Type Word` being decomposed into its three primary subfunctions: `Type Key`, `Predict Word` and `Select Word Prediction`.
(5) Identify any latent parameters, in particular strategies the user may use when carrying out the function. For example, a user may choose to type out each word fully and ignore any word predictions. Alternatively, a user may choose to consistently type as few letters as possible and rely on word predictions as much as possible. Both of these extremes are possible user strategies that can be represented with parameters.
(6) Generate and analyze envelopes. Having identified all key parameters we construct a computational model of system outcomes (such as entry rate) and study how the parameters we have extracted from the function structure model affect system outcomes.

Figure 2 (b) illustrates a decomposition of the overall system function `Type Word` into its three main functions: `Type Key`, `Predict Word` and `Select Word Prediction`. The system outputs a word by either the user typing a series of keys or by the user selecting a word prediction. For clarity, this distinction is not indicated in the diagram, which is why the signal *Word* originates from the overall system function.

To enable the use of word prediction, the `Type Key` function must use a *Key Press* signal to generate an *Observation* signal which is fed into the function `Predict Word`. `Predict Word` then uses a *Language Context* signal to generate a *Word Hypotheses* signal, which can be used by the `Select Word Prediction` function along with a *Word Selection* signal to determine the user's desired word prediction.

The function structures in Figure 2 are not coupled to any particular implementation. For example, consider the function `Type Key`. It can be implemented using a wide range of function carriers, ranging from choices of hardware (physical keys, touchscreen, mid-air, etc.) and algorithms, such as identifying the key by nearest neighbor or by using a probabilistic touch model to infer a probability distribution over all possible keys.

The function structures in Figure 2 are simple but surprisingly versatile. Consider three common computational techniques for word prediction. A simple prefix-tree approach would only require the letter of the detected key and the *Language Context* signal can be provided by a prefix-tree [6] trained on a vocabulary of words. A statistical token-passing decoder [30] requires `Type Key` to output an observation that contains a probability distribution over all keys of the keyboard and the *Language Context* signal would be provided by long-span statistical language models over characters and words. The statistical decoder would then use the likelihood distribution provided by `Type Key` and the language model priors provided by the *Language Context* signal to perform a token-passing search in `Predict Word` and arrive at a posterior probability distribution over all words in the vocabulary. A neural network architecture (e.g. [10]) would not have this separation of prior and likelihood models but instead subsume them into an integrated decoding process in `Predict Word`.

Despite the high-level function structure and having made no commitment to any function carrier solutions, we can still parameterize the function structure model in Figure 2 (b) for the purpose of understanding the utility of word predictions from a system design perspective. Importantly we make no claim to explicitly model the user. From such a system perspective, the user is represented as merely two distinct signals to the overall function `Type Word`, suggesting two fundamental parameters: 1) the time it takes the user to type the desired key—T_{key}; and 2) the time it takes the user to select a predicted word—T_{react}. A third parameter is the prediction accuracy of `Predict Word`—P_{pred}. In addition, there is a latent set of parameters captured by the user's *typing strategy*, which is a set of rules for deciding whether to type a key or select a word prediction. We will explain and define all these parameters in the next section when we present the computational model.

It is also possible to distill additional, more fine-grained, parameters, such as the number of predicted word alternatives the user has to choose from, and the noise characteristics of the user typing a key or selecting a word prediction. However, these parameters can, for the purpose of a system design analysis of the merits of word predictions, be subsumed into the previously introduced parameters, which we will explain when we introduce the computational model. In addition, it is possible to parameterize a language model's influence on word prediction accuracy. As we will demonstrate, this influence can be modeled using a construct we call a *stochastic oracle*. We will revisit parameter selection for envelope analysis later in this paper when we discuss the further implications of using function models in text entry research.

5 PREDICTIVE TEXT ENTRY

We will in this section evolve the parameterized function structure for predictive text entry to fully demonstrate the potential of using parameterized function structure models and envelope analyis in text entry system design.

5.1 Computational Model

Following the parameterization of the function model, fundamentally, the process of entering text on a predictive interface can be broken down into two main components: pressing keys and responding to, and possibly selecting, presented word predictions (sometimes called suggestions). Typing is essentially a varied sequence of these two fundamental actions. These interface actions are followed according to some *strategy*, this is a decision policy for choosing among a fixed number of user interface actions. Each action in the sequence takes some amount of time. It is to be expected that the cognitive task T_{react}, takes somewhat longer than the time to move to and press a key T_{key}. The overall text entry rate is thereby a function of these parameters and the chosen *strategy*. A flowchart representation of the text entry process is shown in Figure 3, which is explained in detail in section 5.2.

The prospective benefit of predictive keyboards is that word completion suggestions can potentially save some of these key presses, T_{key}. Performance is therefore also a function of the language model and its ability to make the correct predictions. In its simplest form this can be represented by the term P_{pred}: the probability of the language model generating the correct prediction.

5.1.1 Stochastic Oracle. We here introduce a method that allows the prediction probability, P_{pred} of the language model to be controlled. It relies on the fact that the primary procedure of evaluating performance of text entry methods is through a transcription task, in which participants are asked to copy stimulus phrases from a carefully selected phrase set. Because the phrases being typed are pre-defined, the system can predict with 100% certainty what the user is attempting to type. The theoretical limit of 100% prediction accuracy can thereby be emulated—this is known as the *oracle*. This oracle can be configured such that it sometimes withholds the perfect predictions and instead obtain predictions from a conventional language model.

By combining the oracle and a conventional language model we can control the proportion in which they act (defined by the ratio R_{blend}). This allows a range of P_{pred} values to be achieved, where the upper bound on P_{pred} is 100% and the lower bound is set by the language model. We call this blended model the *stochastic oracle*. Whenever the system demands word predictions, a Bernoulli trial is carried out which decides whether the prediction will come from the oracle or the language model. The overall performance of the blended language model is measured by the metric P_{pred}—prediction accuracy, which is the proportion of queries that yielded correct predictions:

$$P_{pred} = \frac{N_{success}}{N_{success} + N_{fail}}, \tag{1}$$

where $N_{success}$ is the total number of correct predictions and N_{fail} is the total number of incorrect predictions recorded during the simulation of the phrase set.

5.1.2 Phrase Set and Language Model. Transcription-based evaluation techniques are by far the most common throughout the literature, as these have inherently strong internal validity. To also ensure good external validity, we use the Enron mobile phrase set [29], which is based on a large collection of emails typed on mobile devices. The style and language of the content should therefore be a good representation of mobile text.

The simulations use the language model implementation from the *BerkeleyLM* [23] open-source Java library, in conjunction with a prefix-tree, to implement word predictions. We use an *n*-gram language model built from a publicly available model, derived from a crowdsourced message data set [28].

It is important to emphasize, however, that the exact choice of language model is not critical in this approach, owing to the stochastic oracle method described earlier. Nevertheless, this does provide a sensible baseline from which to start our analysis.

5.1.3 Baseline Conditions. T_{key} and T_{react} parameters that are representative of typing on mobile phones can be obtained from prior studies on Fitts' law and the Hick-Hyman law. Using parameter values obtained from the literature, we obtain values of $T_{key} = 0.26$ (based on [4]) and $T_{react} = 0.45$ (based on [16] and [5]). These values serve as a *baseline* for the analysis. The achievable entry rate will depend on these parameters and, specifically, their relative values are expected to impact the results. The model is designed with this in mind, allowing for sensitivity analysis of the simulation results over any desired range of the parameters T_{key} and T_{react}.

5.2 Strategies

The *typing strategy* of the user is determined by three parameters. These can be combined in a number of ways to capture a broad range of typing strategies.

5.2.1 Minimum Word Length — L_{min}. The minimum word length strategy restricts the use of predictions to only words above a certain length, L_{min}. The intuition behind this is that the user is able to minimize distractions incurred by word predictions, by only considering predictions for longer words; for which the potential keystroke savings are greatest.

5.2.2 Type-then-Look — k_{look}. When typing a word, the likelihood of a correct prediction increases with each new letter entered, as this narrows down the search space for the language model. This parameter defines how many letters the user types of a word before looking at any word predictions. By holding off on the use of predictions early in the typing of a word, such strategies sacrifice potential keystroke savings for increased reliability of predictions.

5.2.3 Perseverance — p_{max}. When typing a word, the user is unlikely to pursue word predictions indefinitely. The model assumes that predictions are considered at least once, but if the correct prediction is not obtained by the *n*th letter, predictions are abandoned and the word is typed out normally. This cut-off is defined by the parameter p_{max}.

Given a set of parameters and a phrase to transcribe, the simulator goes through the process of typing the phrase letter-by-letter, and, where appropriate, uses predictions obtained from a language model. The steps in the simulator are broken down as a flowchart in Figure 3. The simulator keeps track of the number of T_{key} and T_{react} events throughout, as well as keystroke savings for each typed word and the success and failure counts whenever predictions are consulted. The entry rate is calculated from the summation of these time events and the number of characters in the typed phrase. This process is carried out for each phrase in the set and finally the simulator calculates the average entry rate and standard deviation over the phrase set.

We assume users are using their best effort to type each intended key. However, we do not assume error-free typing as the degree of error correction assistance by the system is subsumed in the previously defined parameter P_{pred}.

We calculate keystroke savings, KS, as a percentage, based on the following definition:

$$KS = \left(1 - \frac{\alpha}{\beta}\right) \times 100\%, \tag{2}$$

where α is the number of keystrokes the user has to type when assisted with word predictions and β is the number of keystrokes the user has to type without word prediction assistance. A higher keystroke savings is better. The two extreme cases are either perfect word prediction with no demand for the user to type a single key (in which case $\frac{\alpha}{\beta} = 0$ and the keystroke savings are 100%) and completely unsuccessful word prediction (in which case $\frac{\alpha}{\beta} = 1$ and the keystroke savings are 0%). KS is recorded over a single phrase. The average KS is the mean of all KS recorded over the full phrase set. This is the measure used in Figure 6.

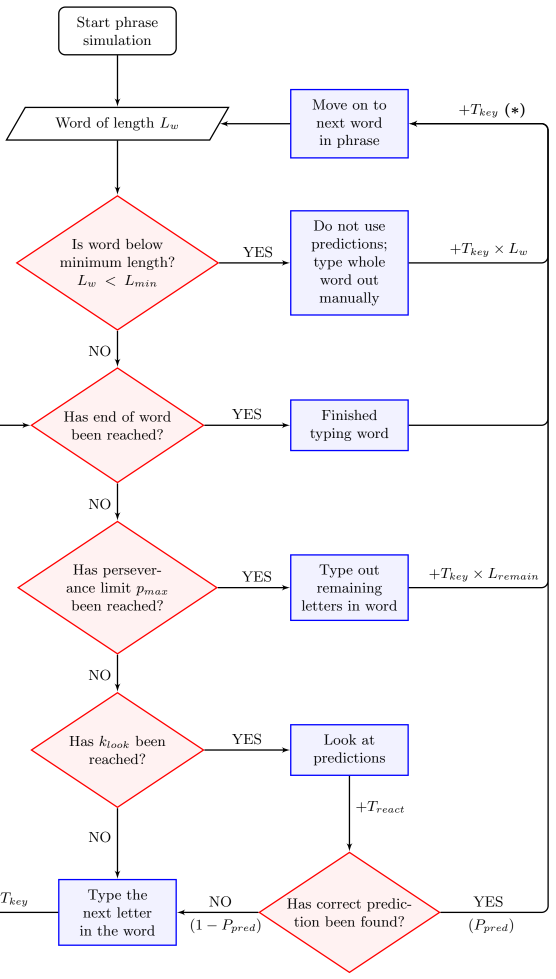
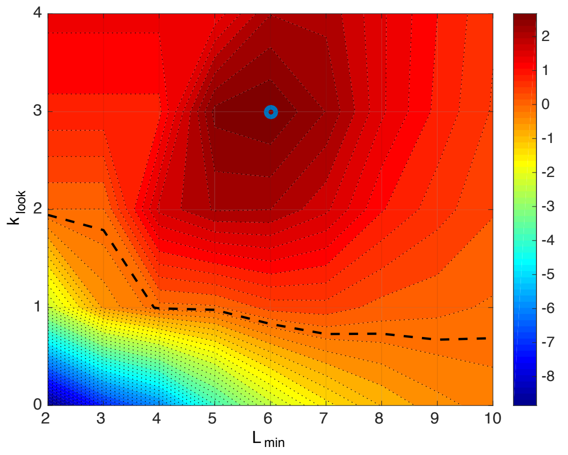

Start phrase simulation
Word of length L_w
Move on to next word in phrase
$+T_{key}$ (∗)
Is word below minimum length? $L_w < L_{min}$
YES
Do not use predictions; type whole word out manually
$+T_{key} \times L_w$
NO
Has end of word been reached?
YES
Finished typing word
NO
Has perseverance limit p_{max} been reached?
YES
Type out remaining letters in word
$+T_{key} \times L_{remain}$
NO
Has k_{look} been reached?
YES
Look at predictions
$+T_{react}$
NO
Type the next letter in the word
$+T_{key}$
NO $(1 - P_{pred})$
Has correct prediction been found?
YES (P_{pred})

Figure 3: Flowchart showing the step-by-step process of simulating typing a phrase. Note (∗): The extra T_{key} event before the start of each new word is because the user has to type a space (or punctuation) before moving on to the next word, the same T_{key} event also applies when a prediction is found as it has to be selected by the user.

5.3 Results

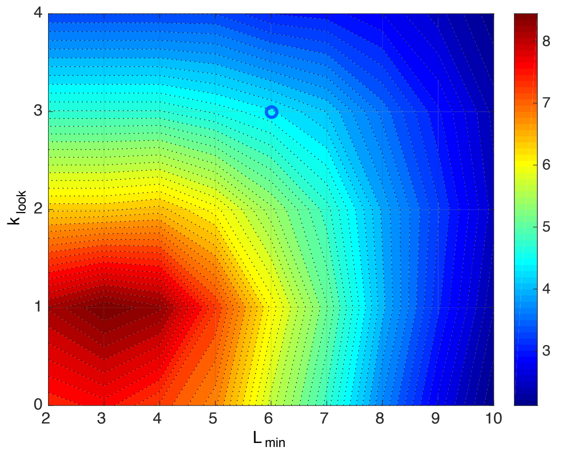
The motivation behind this work is to identify under what conditions word predictions provide an advantage over straightforward typing. Net entry rate is therefore used as the primary performance metric, which is the difference between the entry rate achieved using predictions and the entry rate for unassisted typing. Entry rate is measured in words per minute and uses the convention that one word is five characters including spaces. The standard deviation of the entry rate provides an indicator of the reliability of a particular typing strategy; the smaller the standard deviation the more consistent the performance.

Simulations of the full phrase set were run for different combinations of the strategy parameters: L_{min} ranging from 2 to 10, k_{look}



Parameter	Role	Value
L_{min}	variable	2–10
k_{look}	variable	0–4
p_{max}	fixed	5
T_{key}	fixed	0.26
T_{react}	fixed	0.45

Figure 4: Net entry rate (difference between predictive and unassisted text entry rate) in words per minute plotted over a range of strategies, defined by the parameters L_{min} (minimum word length) and k_{look} (number of letters typed before looking at predictions). Other parameters are fixed at the values shown in the table. The red hot region indicates high net positive entry rate and the cool blue region represents large net negative entry rate. The black dashed line marks the zero-crossing, above which word predictions provide a performance gain. The deeper the red the higher the net entry rate. Under these conditions the highest net entry rate is attained with $L_{min} = 6$, $k_{look} = 3$.



Parameter	Role	Value
L_{min}	variable	2–10
k_{look}	variable	0–4
p_{max}	fixed	5
T_{key}	fixed	0.26
T_{react}	fixed	0.45

Figure 5: Standard deviation of net entry rate across the phrase set in words per minute plotted over a range of strategies, defined by the parameters L_{min} (minimum word length) and k_{look} (number of letters typed before looking at predictions). The ranges of strategy parameters and the fixed parameter values are the same as in Figure 4. The red hot color indicates high standard deviation and the cool blue color represents low standard deviation. A lower standard deviation is desirable as this indicates a more consistent level of performance. Typing strategies that make extensive use of predictions (small L_{min} and k_{look}) have a higher standard deviation than more conservative strategies.

from 0 to 4 and p_{max} from 1 to 5. This covers a broad range of conceivable approaches. The results of the simulation are values of the mean and standard deviation of entry rate as a function of these three strategy parameters and the parameters T_{key} and T_{react}.

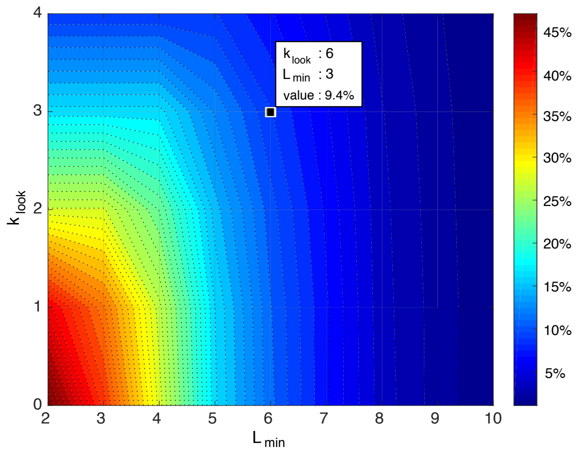
The perseverance p_{max} is found to have little impact on the results for values of $p_{max} > 2$, as the majority of words were predicted correctly within the first two to three letters. Therefore most strategies, regardless of p_{max}, showed very similar results, particularly for strategies where $k_{look} > 1$, which showed a shift in entry rate of only 1.42 words per minute in the most extreme case and generally much lower than this across the parameter space.

The entry rate, however, is strongly dependent on the choice of minimum word length L_{min} and type-then-look k_{look} parameters, which showed shifts of 7.0 and 3.6 words per minute respectively. Most of the information of interest in the dataset is therefore only in two of the three parameters. To visualize this data we can fix the p_{max} parameter and plot the results as a heatmap with coordinates defined by the two strategy parameters: (L_{min}, k_{look}). The resulting maps (Figures 4–7) give an insight into how the parameters are related and identify conditions under which predictive text may be beneficial, as well as where the limitations are. The data in these figures is obtained with $p_{max} = 5$ and time-cost parameters fixed at their *baseline* values: $T_{key} = 0.26$ and $T_{react} = 0.45$ seconds.

The map of net entry rate in Figure 4 illustrates that performance is strongly tied to the choice of typing strategy. The black dashed line in Figure 4 is the zero-crossing and marks the boundary between performance gain and loss due to predictions. In general, any strategy that looks at word predictions without typing at least one letter, i.e. $k_{look} = 0$, is shown to slow the user down. Performance gains are only realized if $k_{look} \geq 2$, that is, the user types at least the first two letters of each word before looking at predictions. The reliability of predictions increases with the number of letters typed.

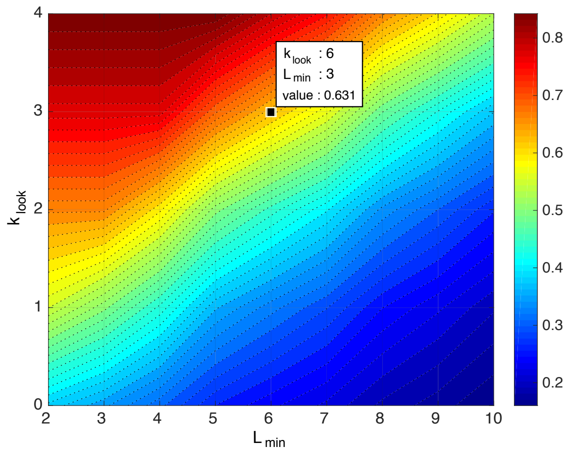
The net entry rate can range from −8.8 to +2.9 words per minute solely as result of the choice of typing strategy. The highest point identifies the optimal typing strategy: $L_{min} = 6$, $k_{look} = 3$. This point is marked in Figures 4–7.

Looking at the map of standard deviation in Figure 5, the highest values are found when L_{min} and k_{look} are small. The language



Parameter	Role	Value
L_{min}	variable	2–10
k_{look}	variable	0–4
p_{max}	fixed	5
T_{key}	not applicable	—
T_{react}	not applicable	—

Figure 6: Average keystroke savings plotted over a range strategies, defined by the parameters L_{min} (minimum word length) and k_{look} (number of letters typed before looking at predictions). The same ranges of typing strategy parameters are used as in Figures 4 and 5. Note that the keystroke savings measure is independent of T_{key} and T_{react}. The typing strategy that achieved the highest net entry from Figure 4 is highlighted, for which the average keystroke savings is only 9.4% per word. The highest keystroke savings are towards the bottom left side of the graph, with average keystroke savings of over 40% per word. However, this region correspond to a net negative entry rate in Figure 4. This highlights the lack of correlation between keystroke savings and entry rate.



Parameter	Role	Value
L_{min}	variable	2–10
k_{look}	variable	0–4
p_{max}	fixed	5
T_{key}	not applicable	—
T_{react}	not applicable	—

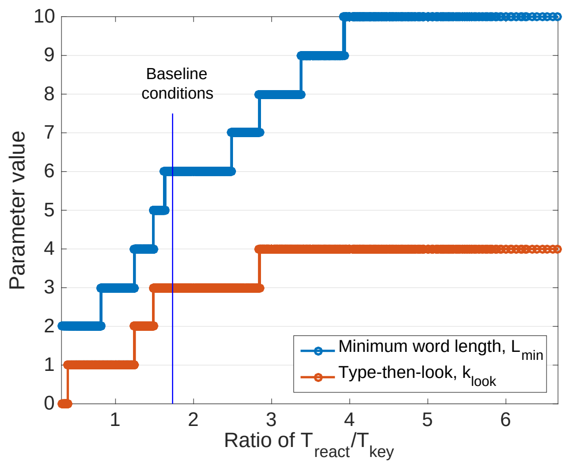
Figure 7: Success rate of predictions plotted over a range strategies, defined by the parameters L_{min} (minimum word length) and k_{look} (number of letters typed before looking at predictions). The same parameter ranges are used as in Figures 4 and 5. Whenever the user consults the predictions, successful and unsuccessful outcomes are tallied. The success rate is simply the proportion of outcomes that were successful, as defined in Equation 1. As expected, higher values of k_{look} have higher prediction success rate, as more information is available to the language model before it has to make a prediction. Higher values of L_{min} lead to lower success rates as the language model is expected to predict longer, likely less frequently occurring words.

model is a source of variance, therefore the greater the reliance on the predictions, the more of this variance is reflected in the resulting entry rates. Small values for minimum word length L_{min} mean that predictions are used for a larger proportion of words, hence the increased standard deviation. The value of k_{look} sets how many letters of a word are typed before predictions are considered. The reliability of predictions increases with the number of letters typed, hence standard deviation decreases as k_{look} increases. A low standard deviation is desirable as this implies a more consistent typing experience, which is less likely to frustrate the user. This objective has to be balanced with the desire for a high net entry rate. Fortunately, from comparing Figures 4 and 5, these are not conflicting objectives.

Comparing the pattern of keystroke savings in Figure 6 to that of entry rate in Figure 4 indicates that strategies that attempt to maximize keystroke savings do not necessarily translate well into increased performance (as measured by entry rate). In fact, to the contrary, it suggests that pursuing such a policy is detrimental to performance.

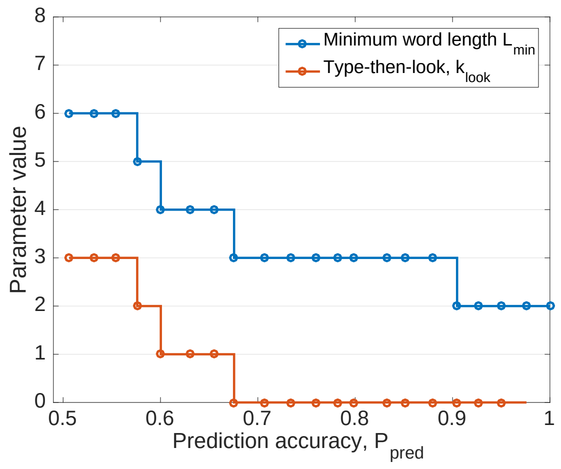
As expected, Figure 7 shows that the rate of success of using predictions improves as the k_{look} parameter increases. Looking at the gradient in the L_{min} axis shows that success rate decreases with L_{min}, as longer words are less frequent and therefore harder to predict for the language model which, because it is based on a mobile device phrase set, favors predicting short, simple words.

5.3.1 Sensitivity Analysis. The results in Figures 4 through 7 are only for the baseline time-cost conditions. The optimal strategy identified in Figure 4 is therefore only optimal for the given baseline time-cost parameters and base language model. The optimal strategy is likely to be different depending on the relative magnitude of the physical and cognitive constraints, T_{key} and T_{react}. These constraints depend on aspects of the interface as well as the motor ability and experience level of the user. To see how the optimal



Parameter	Role	Value
L_{min}	observed	–
k_{look}	observed	–
p_{max}	observed	–
T_{key}	variable	0.15–0.65
T_{react}	variable	0.2–1.0

Figure 8: A plot revealing how the optimal typing strategy changes as a function of the ratio of the cognitive and physical parameters, T_{react}/T_{key}. The baseline condition is marked $T_{react}/T_{key} = 0.45/0.26 = 1.73$. The results are from simulations without the stochastic oracle. The graph shows that as the ratio increases the optimal strategy shifts towards restricting to longer words (L_{min} increases) and providing more characters before checking predictions (k_{look} increases). This is because the larger the ratio the more T_{react} dominates the typing process and therefore the greater the relative cost of attending to predictions.



Parameter	Role	Value
L_{min}	observed	–
k_{look}	observed	–
p_{max}	observed	–
T_{key}	fixed	0.26
T_{react}	fixed	0.45

Figure 9: A plot revealing how the optimal values for the typing strategy parameters L_{min} and k_{look} change as a function of the prediction accuracy P_{pred} (as defined in Equation 1) achieved by the blended language model. The plotted points correspond to blend ratio, R_{blend}, values ranging from 0 to 1 in increments of 0.05. The R_{blend} values themselves are not of interest, but rather the prediction accuracy P_{pred} of the resulting blended language models; hence P_{pred} is chosen as the x-axis. The first data point, at around $P_{pred} = 0.5$, is for the *base* language model ($R_{blend} = 0$). These results are all under *baseline* conditions: $T_{key} = 0.26$ and $T_{react} = 0.45$.

strategy changes under different T_{key} and T_{react} constraints we carried out a sensitivity analysis.

Across the same range of typing strategies (L_{min}: 2–10; k_{look}: 0–4; p_{max}: 1–5) entry rate is calculated for different values of the constraints T_{key}: 0.15–0.65 and T_{react}: 0.2–1.0. For each combination of T_{key} and T_{react} we recorded the highest performing typing strategy and calculated the ratio T_{react}/T_{key}. The results of this analysis are plotted in Figure 8, which shows how the optimal typing strategy parameters change as a function of the ratio of the time-cost parameters.

5.3.2 Stochastic Oracle Results. A series of simulations were run over the same range of parameters as before (L_{min}: 2–10; k_{look}: 0–4; p_{max}: 1–5), using *baseline* conditions, but for a range of oracle blends—R_{blend} in a range from 0 to 1 in increments of 0.05, where $R_{blend} = 0$ is the base language model only, $R_{blend} = 1$ is the oracle only, and intermediate values are a mixture of both. Similar to the sensitivity analysis, the optimal typing strategy (highest entry rate) was recorded as a function of R_{blend}. The prediction performance P_{pred} was calculated using Equation 1. The results of this analysis are plotted in Figure 9. Note that the x-axis is not of the oracle blend ratio R_{blend}, but of the prediction accuracy, P_{pred}. Smaller values of L_{min} mean that predictions are used more frequently. As expected L_{min} decreases as prediction accuracy increases. k_{look} also decreases because the increased reliability of predictions means that, on average, fewer letters are needed before the correct predictions are found.

5.4 Discussion

Prior work has looked at modeling different aspects related to mobile text entry in considerable detail: the physical process of typing on touchscreen keyboards [3, 4, 26] and the performance of text predictions measured through keystroke savings [7, 9]. These studies, however, have mostly been done in isolation. There are few studies that model text entry with word predictions in a complete context [12, 15, 20, 25]. These only explore a limited scope of typing strategies and none of these account for the influence of the language model.

We have used a parameterized function structure model of predictive text entry to build a comprehensive model of the predictive text entry process that accommodates a wide range of typing strategies, as well as a mechanism for controlling the performance of the language model. We observe that the typing strategy adopted by the user plays a critical role in realizing the full potential of a predictive text entry system. For the *baseline* constraints the optimal strategy identified is to use word predictions only for words of at least six letters in length ($L_{min} = 6$) and to type out the first three letters of a word before attending to predictions ($k_{look} = 3$).

We found that keystroke savings as a performance measure does not correlate well with text entry rate. In particular, the relative magnitude of physical and cognitive costs, described by the ratio T_{react}/T_{key}, was found to govern the optimal strategy, with higher ratios generally favoring strategies that are more reserved in their use of predictions.

The results from the stochastic oracle simulations substantiate the feasibility of using an oracle blended with a language model as means of systematically controlling the effect of prediction quality.

5.5 Design Implications

Some general design implications for text entry systems can be derived from the findings presented here. When designing a text entry system a useful starting point is to estimate the ratio of the physical and cognitive cost parameters T_{react}/T_{key} of the system. These parameter values can be obtained from published Fitts' law and Hick-Hyman law models, as outlined previously in this paper, or by empirical studies. The T_{react}/T_{key} ratio provides a baseline operating point from which to begin analyzing the cost-benefit trade-offs in the system.

A plot of performance envelopes, such as Figure 4, can then help identify both typing strategies that result in performance loss and should be avoided, as well as typing strategies that are theoretically optimal. Whilst changing the user's typing behavior directly might not be possible, it is plausible that the user can be steered away from poor typing behavior and towards adopting more favorable strategies by adjusting the behavior of the typing interface with these performance boundaries in mind.

For example, from Figure 4 we find performance gains are only realized when typing at least two letters before looking at predictions ($k_{look} \geq 2$). The text entry system could be made to withhold predictions completely until at least the first two letters of a word are typed. This gives rise to another question: should a predictive keyboard show completion suggestions for one-, two- or three-letter words at all when the cognitive cost of attending to the prediction outweighs the benefits of keystroke savings for these words? Similarly, the analysis of the optimal typing strategy as a function of T_{react}/T_{key}, as in Figure 8, revealed that for any ratio above 1, the minimum word length (L_{min}) should be at least 3, and at the baseline conditions this rises to 6 letters. It is worth considering biasing word predictions in favour of words ≥ 6 letters to try to push the user's operating point to higher values of L_{min}.

5.6 Function Models in Text Entry

In addition to assisting system design of predictive text by providing computational modeling results, this work has highlighted how developing high-level parameterized function models can assist text entry system design in two ways: 1) a *structural* design contribution, which means an established function structure is redesigned; and 2) a *parameter analysis* design contribution, which means a function structure is parameterized and the designer analyzes the effects of the controllable and uncontrollable parameters on expected system outcomes.

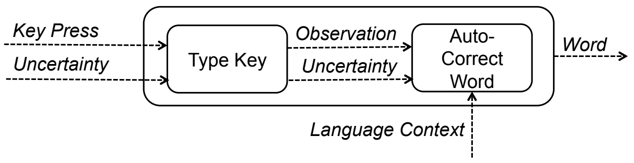


Figure 10: A sketch of an overall system function `Type Auto-Corrected Uncertainty-Regulated Word` decomposed into two main functions: `Type Key` and `Auto-Correct Word`. The dashed arrows indicate signal flow.

As an example of this versatility in early system design analysis, consider the sketch of a function model for the overall function `Type Auto-Corrected Uncertainty-Regulated Word` shown in Figure 10. A well-known problem with auto-correct is the so-called auto-correct trap [31]—the auto-correct algorithm incorrectly replacing text the user intended to type. Weir et al. [31] introduced a solution to this problem that allowed users to communicate their uncertainty in their own key presses to the system by pushing harder on the keys they wanted to prevent the system from auto-correcting.

Figure 10 demonstrates how this uncertainty regulation can be modeled as a function model by connecting one additional *Uncertainty* signal to the `Type Key` function, which is then propagating the *Uncertainty* signal along with the usual *Observation* signal to the function `Auto-Correct Word`. This illustrates a *structural* design contribution—the function structure model is redesigned to incorporate additional functionality.

The function model in Figure 10, already suggests several analyses that can be carried out. First, the medium of the *Uncertainty* signal can be investigated. For example, Weir et al. [31] used pressure while later work used touch duration [27]. Both choices imply more fine-grained function models with functions, such as `Measure Pressure` or `Measure Touch Duration`. Second, the function `Auto-Correct Word` can be decomposed into its own function structure model, detailing concerns such as how user uncertainty around a key press should be modulated by the auto-correct algorithm. For example, integration of uncertainty can be a mixture weight applied when calculating the posterior probability of a word in the decoding process [31], or it can be thresholded [27]. As in this paper, follow-up computational modeling can then be used to comprehensively answer how these and other design parameters affect performance and help guide informed user studies at well understood operating points of the system design space. This illustrates a *parameter analysis* design contribution—the designer parameterizes the function structure and analyzes the effects of the controllable and uncontrollable parameters.

The function model for word prediction in this paper can similarly be extended to explicitly model auto-correct, sentence prediction and the use of interaction context, such as the current location of the text entry system, time-of-day or nature of the text.

6 DISCUSSION

We have explained how parameterized function structures coupled with envelope analysis can guide the design of intelligent text entry systems. We demonstrated this approach with a detailed example illustrating how it can be used to help understand the underlying variables underpinning a word prediction system. In this section we discuss the general implications of this work.

6.1 Using Parameterized Function Structures to Critique User Interface Systems Research

While the approach presented in this paper is intended to support system design, it is also suitable for critiquing user interface system research from two perspectives. First, a system's function structure models and their underpinning controllable and uncontrollable parameters can serve as system contributions in their own right, demonstrating how a new functional design, or a new parameterization of an existing functional design, can result in new emergent user interface system solutions. Examples of such hypothetical contributions are discussed in Section 5.6 for the function structure model illustrated in Figure 10. Second, an existing parameterized function structure model provides insights into controllable and uncontrollable design parameters governing the system. This opens up for a discussion on all these parameters: their importance and influence on system performance, the ability to model them, and scope for investigating their role in the system using quantitative and qualitative methods. This is possible as the approach forces all these design parameters to be explicitly defined as integral parts of the system rather than some of them serving as implicit tacit variables that govern performance but are not explicitly acknowledged as part of the system.

6.2 The Role of Evaluation

The strength of the approach in this paper is a change of perspective: it allows a calculation of a range of operating points for many parameter values and strategies, where each specific parameter set and choice of strategy corresponds to a particular empirical investigation. The purpose of this work is not to predict individual user behavior but to elucidate emergent system behavior of a function structure model as a consequence of variations in controllable and uncontrollable parameters. This results in a emergent map of the system's behavior which can then be used to either: 1) revisit the function structure model or parameterization; 2) consider the effect of different function carriers (for instance, different algorithms or sensors); or 3) empirically study and validate outcomes for particular parameter settings, carefully chosen using envelope analysis as opposed to the de-facto practice of implicitly choosing operating points for user studies by determining levels of the independent variables largely based on intuition instead of by analysis.

6.3 Integration of Parameterized Function Structures in Design Practice

A common design activity, either implicitly or explicitly, is optimization—identifying a set of key parameters in the design and finding their optimal values. Fundamentally, such a process focuses on identifying the most suitable operating point of the system design. In contrast, the approach in this paper visualizes the impact of all operating points emerging from all system configurations. While this approach does require some effort by the designer, it can assist the design process in at least five ways.

First, analyzing the effects of controllable system parameters helps the designer identify design parameters that can be optimized for better performance.

Second, the process itself of identifying key functions and their signals can help the designer in creating a map of the system. Such a *system map* can both assist the designer in considering new designs and help the designer explaining the system to other team members, stakeholders and end-users.

Third, envelope analyses can provide insight into *plausible* operating points in the design and thereby assist the design of user studies. This can save time and effort, as envelope analyses make it easy to identify unpromising operating points in the design and thereby discourage user studies that are unlikely to yield any useful design insights. It can also help deciding on a particular study design as envelope analyses can reveal regions in the design space where the performance impact is particularly large given a few parameter variations, which suggests studying these parameter variations in, for example, a controlled experiment.

Fourth, the approach can assist in eliciting system requirements. Controllable parameters are design parameters the designer can influence and an understanding of achievable performance under certain controllable parameter configurations can guide requirements on their values in the finalized design. For example, the number of word prediction slots in a word prediction interface is a controllable parameter and its effect can be studied in envelope analysis. In contrast, uncontrollable parameters cannot be directly influenced by the designer. An understanding of the system's performance under certain uncontrollable parameter configurations allows sensitivity analysis of the system's performance when influenced by parameters the designer cannot directly govern. Such an understanding may help defining requirements on the robustness of the system when exposed to such parameters. For example, an understanding of an auto-correct function's resilience to typing errors (an uncontrollable parameter) can help set realistic requirements on the system's expected performance when exposed to typing errors.

Fifth, as illustrated by Kristensson et al. [18], envelope analysis can allow the study of the performance potential of a system design that is difficult to assess in user studies, for example, because the system has to be deployed to nonspeaking individuals with motor disabilities for a prolonged amount of time [18].

Finally, while envelope analysis is by its very nature quantitative, an interesting avenue of future work is to investigate whether parameterized function structures can also be used as scaffolding for qualitative research, such as non-directed interviews and observations. In such inquiries it is often helpful for the researcher to *know what to look for* and having a system broken down into essential functions, signals and parameters may help focus such investigations. For example, a qualitative study might focus on reasons why users do not leverage word predictions when the operating points in the envelope analyses are indicating a clear benefit. Hypothetically, such an inquiry might for instance find that some individuals find it stressful to look at word predictions, even though they would be clearly beneficial at their individual operating points.

7 CONCLUSIONS

There are many factors influencing the success or failure of a text entry system, such as a word prediction system. These factors are difficult to control for experimentally and, consequently, drawing conclusions about the design implications of the empirical results is often an unreliable process. Further, such studies lack a theoretical common ground, which makes comparisons between studies impractical or impossible. In contrast to empirical studies, the strength of the modeling approach in this paper is a change of perspective: we calculate a range of operating points for many parameter values and strategies, where each specific parameter set and choice of strategy would correspond to a condition in an empirical study. These calculations are carried out by a computational model underpinned by a parameterized function structure model that at a high-level identifies the key design parameters of the system.

This paper has illustrated this approach by introducing a parameterized function structure model of predictive text entry, which allowed us to derive a computational model that enabled a systematic exploration of the design space of predictive text entry. The approach taken in this paper complements traditional text entry system design techniques and can assist designers in identifying and justifying operating points for designs. Once an operating point has been decided, it can be further explored in quantitative and qualitative user studies. In this way, function models and computational models provide scaffolding for better motivating the system design context of user studies and can help explore areas of the design that are poorly understood.

ACKNOWLEDGMENTS

This work was supported by EPSRC (grants EP/R004471/1 and EP/S027432/1).

REFERENCES

[1] Ohoud Alharbi, Wolfgang Stuerzlinger, and Felix Putze. 2020. The effects of predictive features of mobile keyboards on text entry speed and errors. *Proceedings of the ACM on Human-Computer Interaction* 4, ISS, Article 183 (2020), 16 pages. https://doi.org/10.1145/3427311

[2] Kenneth C Arnold, Krzysztof Z. Gajos, and Adam T. Kalai. 2016. On suggesting phrases vs. predicting words for mobile text composition. In *Proceedings of the 29th Annual ACM Symposium on User Interface Software and Technology.* 603–608. https://doi.org/10.1145/2984511.2984584

[3] Shiri Azenkot and Shumin Zhai. 2012. Touch behavior with different postures on soft smartphone keyboards. In *Proceedings of ACM MobileHCI.* 251–260. https://doi.org/10.1145/2371574.2371612

[4] Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts law: modeling finger touch with Fitts' law. In *Proceedings of the 31st ACM Conference on Human Factors in Computing Systems.* 1363–1372. https://doi.org/10.1145/2470654.2466180

[5] Andy Cockburn, Carl Gutwin, and Saul Greenberg. 2007. A predictive model of menu performance. In *Proceedings of the 25th ACM Conference on Human Factors in Computing Systems.* 627–636. https://doi.org/10.1145/1240624.1240723

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms.* MIT press.

[7] Andrew Fowler, Kurt Partridge, Ciprian Chelba, Xiaojun Bi, Tom Ouyang, and Shumin Zhai. 2015. Effects of language modeling and its personalization on touchscreen typing performance. In *Proceedings of the 33rd ACM Conference on Human Factors in Computing Systems.* 649–658. https://doi.org/10.1145/2702123.2702503

[8] Nestor Garay-Vitoria and Julio Abascal. 2006. Text prediction systems: a survey. *Universal Access in the Information Society* 4, 3 (2006), 188–203. https://doi.org/10.1007/s10209-005-0005-9

[9] Nestor Garay-Vitoria and Julio Abascal. 2010. Modelling text prediction systems in low- and high-inflected languages. *Computer Speech and Language* 24, 2 (2010), 117–135. https://doi.org/10.1016/j.csl.2009.03.008

[10] Shaona Ghosh and Per Ola Kristensson. 2017. Neural networks for text correction and completion in keyboard decoding. *arXiv:1709.06429* (2017).

[11] Saul Greenberg and Bill Buxton. 2008. Usability evaluation considered harmful (some of the time). In *Proceedings of the 26th ACM Conference on Human Factors in Computing Systems.* 111–120. https://doi.org/10.1145/1357054.1357074

[12] Heidi Horstmann Koester and Simon P. Levine. 1994. Modeling the speed of text entry with a word prediction interface. *IEEE Transactions on Rehabilitation Engineering* 2, 3 (1994), 177–187. https://doi.org/10.1109/86.331567

[13] Heidi Horstmann Koester and Simon P Levine. 1995. Validating quantitative models of user performance with word prediction systems. In *Proceedings of the 18th Annual Conference on Rehabilitation Technology.* 127–129.

[14] Heidi Horstmann Koester and Simon P. Levine. 1996. Effect of a word prediction feature on user performance. *Augmentative and Alternative Communication* 12, 3 (1996), 155–168. https://doi.org/10.1080/07434619612331277608

[15] Heidi Horstmann Koester and Simon P. Levine. 1998. Model simulations of user performance with word prediction. *Augmentative and Alternative Communication* 14, 1 (1998), 25–35. https://doi.org/10.1080/07434619812331278176

[16] Ray Hyman. 1953. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology* 45, 3 (1953), 188–196. https://doi.org/10.1037/h0056940

[17] Per Ola Kristensson. 2009. Five challenges for intelligent text entry methods. *AI Magazine* 30, 4 (2009), 85–94. https://doi.org/10.1609/aimag.v30i4.2269

[18] Per Ola Kristensson, James Lilley, Rolf Black, and Annalu Waller. 2020. A design engineering approach for quantitatively exploring context-aware sentence retrieval for nonspeaking individuals with motor disabilities. In *Proceedings of the 38th ACM Conference on Human Factors in Computing Systems.* Paper 398. https://doi.org/10.1145/3313831.3376525

[19] Henry Lieberman. 2003. The tyranny of evaluation. *CHI Fringe* (2003).

[20] Ying Liu and Kari-Jouko Räihä. 2010. Predicting Chinese text entry speeds on mobile phones. In *Proceedings of the 28th ACM Conference on Human Factors in Computing Systems.* 2183–2192. https://doi.org/10.1145/1753326.1753657

[21] Dan R. Olsen Jr. 2007. Evaluating user interface systems research. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology.* 251–258. https://doi.org/10.1145/1294211.1294256

[22] Gerhard Pahl and Wolfgang Beitz. 2013. *Engineering Design.* Springer.

[23] Adam Pauls and Dan Klein. 2011. Faster and smaller n-gram language models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics.* 258–267.

[24] Philip Quinn and Shumin Zhai. 2016. A cost-benefit study of text entry suggestion interaction. In *Proceedings of the 34th ACM Conference on Human Factors in Computing Systems.* 83–88. https://doi.org/10.1145/2858036.2858305

[25] Miika Silfverberg, I. Scott MacKenzie, and Panu Korhonen. 2000. Predicting text entry speed on mobile phones. In *Proceedings of the 18th ACM Conference on Human Factors in Computing Systems.* 9–16. https://doi.org/10.1145/332040.332044

[26] William R. 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 (1995), 370–379. https://doi.org/10.1080/01449299508914656

[27] Keith Vertanen, Dylan Gaines, Crystal Fletcher, Alex M. Stanage, Robbie Watling, and Per Ola Kristensson. 2019. VelociWatch: designing and evaluating a virtual keyboard for the input of challenging text. In *Proceedings of the 37th ACM Conference on Human Factors in Computing Systems.* Paper 591. https://doi.org/10.1145/3290605.3300821

[28] Keith Vertanen and Per Ola Kristensson. 2011. The imagination of crowds: conversational AAC language modeling using crowdsourcing and large data sources. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing.* 700–711.

[29] Keith Vertanen and Per Ola Kristensson. 2011. A versatile dataset for text entry evaluations based on genuine mobile emails. In *Proceedings of ACM MobileHCI.* 295–298. https://doi.org/10.1145/2037373.2037418

[30] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: investigating fast mobile text entry using sentence-based decoding of touchscreen keyboard input. In *Proceedings of the 33rd ACM Conference on Human Factors in Computing Systems.* 659–668. https://doi.org/10.1145/2702123.2702135

[31] Daryl Weir, Henning Pohl, Simon Rogers, Keith Vertanen, and Per Ola Kristensson. 2014. Uncertain text entry on mobile devices. In *Proceedings of the 32nd ACM Conference on Human Factors in Computing Systems.* 2307–2316. https://doi.org/10.1145/2556288.2557412