Accelerating Text Communication via Abbreviated Sentence Input

Jiban Adhikary[1] **Jamie Berger**[2] **Keith Vertanen**[1]
jiban@mtu.edu jamieberger16@gmail.com vertanen@mtu.edu
[1]Michigan Technological University, Houghton, Michigan, USA
[2]Washington Leadership Academy, Washington, DC, USA

Abstract

Typing every character in a text message may require more time or effort than strictly necessary. Skipping spaces or other characters may be able to speed input and reduce a user's physical input effort. This can be particularly important for people with motor impairments. In a large crowdsourced study, we found workers frequently abbreviated text by omitting mid-word vowels. We designed a recognizer optimized for expanding noisy abbreviated input where users often omit spaces and mid-word vowels. We show using neural language models for selecting conversational-style training text and for rescoring the recognizer's n-best sentences improved accuracy. On noisy touchscreen data collected from hundreds of users, we found accurate abbreviated input was possible even if a third of characters was omitted. Finally, in a study where users had to dwell for a second on each key, sentence abbreviated input was competitive with a conventional keyboard with word predictions. After practice, users wrote abbreviated sentences at 9.6 words-per-minute versus word input at 9.9 words-per-minute.

1 Introduction

Experienced desktop and touchscreen typists can often achieve fast and accurate text input by simply typing all the characters in their desired text. However, for some users, such quick and precise input is difficult due to a motor disability. Such users may use a virtual touchscreen keyboard, but their touch locations may be slow and inaccurate, e.g. people with Cerebral palsy. Other users may need to click keys by pointing at them with a head- or eye-tracker and dwelling for a fixed time, e.g. people with amyotrophic lateral sclerosis (ALS).

When a person's typing is slow or inaccurate, word completions may provide more efficient input. Word completions predict the most probable words based on the current typed prefix. However, monitoring predictions carries a cognitive cost and may not always improve performance (Trnka et al., 2009). Further, monitoring predictions can be difficult without visual feedback. Eyes-free text input can be slow for users who are visually-impaired (Nicolau et al., 2019), and even slower for users who are motor- and visually-impaired (Nel et al., 2019). Finally, eyes-free text input may be needed in future augmented reality (AR) interfaces where visual feedback is limited or non-existent (e.g. due to lighting or device limitations). In audio-only AR, it is still possible to type on an invisible virtual keyboard (Vertanen et al., 2013; Zhu et al., 2018).

All these cases motivate our interest in exploring alternatives to conventional word completion. Here we investigate accelerating input by allowing users to skip typing spaces and mid-word vowels. We decided to abbreviate in this manner based on past results on touchscreen text input without spaces (Vertanen et al., 2015, 2018), and a study we present here in which 200 people abbreviated email messages. Our interaction approach of abbreviation is similar to features in commercial assistive interfaces (e.g. Grid 3, NuVoice, Lightwriter). Our whole utterance prediction approach is similar to features in touchscreen phone keyboards and in commercial assistive interfaces (e.g. dwell-free sentence input in Tobii Communicator 5).

We modified a probabilistic recognizer to accurately expand abbreviated input by 1) improving our language models by selecting well-matched training data via a neural network, 2) modifying the search to model the insertion of mid-word vowels, and 3) adding a neural language model to the search. We validate our method in computational experiments on over six thousand sentences typed on touchscreen devices. We found that even when 28% of letters were omitted, we recognized sentences with no errors 70% of the time. Selecting

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 6574–6588
August 1–6, 2021.

from the top three sentences, user could obtain their intended sentence 80% of the time.

Finally, we compare word completion and abbreviated sentence input in a user study. In this study, users had to dwell for one second to trigger a tap. We found sentence input was slightly slower than using word completions, but still saved substantial time compared to typing all the characters. Users obtained their desired sentence 68% of the time.

2 Related Work

Abbreviated input. Demasco and McCoy (1992) investigated expanding uninflected words (e.g. "apple eat john") into syntactic sentences (e.g. "the apple is eaten by john"). Gregory et al. (2006) created abbreviation codes (e.g. "rmb" = "remember"). Users selected words from a menu or by typing a code's letters. Typing codes was the most efficient. Pini et al. (2010) detected abbreviated phrases using a Support Vector Machine and expanded them via a Hidden Markov Model (HMM). Their detector and expander were 90% and 95% accurate respectively. Users decreased keystrokes and input time by 32% and 26% respectively.

Shieber and Nelken (2007) allowed users to drop non-initial vowels and repeated consonants. This deleted 26% of the total characters. Using an n-gram word language model and a spelling transducer for each word, they expanded abbreviated text at an error rate of 3.3%. Our work differs in that we: 1) removed spaces between words, 2) did not remove consecutive consonants, 3) used a character language model with no fixed vocabulary.

Tanaka-Ishii et al. (2001) explored Japanese text input with digits. They used an HMM to expand a sequence of digits into characters. Users saved 35% of keystrokes typing on a mobile phone. Han et al. (2009) also used an HMM to expand abbreviations learned from a corpus of Java code. Their approach did not require memorizing abbreviations and provided incremental feedback while typing.

In two studies with 31 users, Willis et al. (2002, 2005) identified common abbreviation behaviors such as vowel deletion, phonetic replacement, and word truncation. They did not release their data and it was on a relatively small number of people. Based on their work, we conducted an abbreviation study with 200 users and also share our data.

Data selection. Mismatch between the training and target text domains can lead to sub-optimal language models. A variety of methods have been developed to address this problem. Lin et al. (1997), Gao et al. (2002), and Yasuda et al. (2008) used language modeling and in-domain perplexity to select training data. In this approach, a language model is trained on a small in-domain dataset. Training instances from an out-of-domain dataset are selected if they are below some perplexity threshold.

Other work has investigated data selection using cross-entropy or cross-entry difference between in- and out-of-domain datasets (Axelrod et al., 2011; Moore and Lewis, 2010; Schwenk et al., 2012; Rousseau, 2013; Mansour et al., 2011; Vertanen and Kristensson, 2011b). In this approach, an in-domain and out-of-domain language models are first trained. Sentences are selected based on a cross-entropy threshold or cross entropy difference calculated from the two language models.

Hildebrand et al. (2005) and Lü et al. (2007) applied information retrieval based techniques to select data. Other method include selecting based on infrequent n-gram occurrences (Gascó et al., 2012; Parcheta et al., 2018), or Levenshtein distance and word vectors (Chinea-Rios et al., 2018).

Duh et al. (2013) employed the data selection method of Axelrod et al. (2011), which builds upon Moore and Lewis (2010)'s approach. The main distinction is that they used neural language models for selection rather than n-gram models. Chen and Huang (2016), Peris et al. (2017), and Chen et al. (2016) selected based on convolutional and bidirectional long short-term memory neural networks.

Bidirectional neural models like BERT (Devlin et al., 2019) has proven effective in many natural language tasks. Ma et al. (2019) used BERT for domain-discriminative data selection. Hur et al. (2020) used BERT for domain adaptation and instance selection for disease classification. Our selection method is similar to these methods but focuses on selecting conversational-style sentences.

Decoding noisy input. Text entry interfaces often use a probabilistic decoder to infer a user's text from time sequence data (Vertanen et al., 2015; Kristensson and Zhai, 2004; Zhai et al., 2002; Zhai and Kristensson, 2008). Typically, a keyboard likelihood model and a language model prior are used to infer a user's text from input with incorrect, missing, or extra characters. To date, these approaches have mostly used n-gram language models.

Ghosh and Kristensson (2017) corrected typos in tweets to a low character error rate of 2.4% by using a character convolutional neural network, an

encoder with gated recurrent units, and a decoder with attention. The twitter typo data contained sequences with a similar number of characters to the target. In our work, we show acceptable character error rate can be achieved on input not only with typos, but also with missing spaces and mid-word vowels. We show the advantage of using a recurrent neural network language model (RNNLM) directly in the decoder's search or to rescore hypotheses.

3 Free-form Abbreviation Study

To better understand how people do free-form abbreviation, we conducted a study on Amazon Mechanical Turk. As a pilot, we had 26 workers abbreviate an email from the Enron mobile data set (Vertanen and Kristensson, 2011a). We designed our instructions based on Willis et al. (2005). Workers abbreviated the same email three times. Each time the worker was asked to abbreviate in three ways: heavily, as little as possible, or as they saw fit.

In our pilot, we found workers abbreviated similarly regardless of instructions. Thus, we designed a single set of instructions for our main study that asked workers to imagine they were using artificially intelligent (AI) software that was good at guessing their intended text from an abbreviated form. They were told to shorten words by removing or changing letters, but they should avoid shortening words that might be hard for the system to guess and that they should not omit words entirely. See the appendix for our instructions. Our supplementary data contains all the data from the study.

We recruited 200 workers who each abbreviated ten emails. In our analysis, we used 1,308 of the 2,000 emails. We filtered out emails that did not have the same number of words as their original emails. This filtering helped us to align the sentences by word. Punctuation was removed except apostrophes and at signs. We lowercased the text.

3.1 Abbreviation Behavior

We found 90% of abbreviated words were an in-order subsets of their full spelling. On average, 21% of a word's letters were deleted. Of these, 16% were consonants and 42% were vowels. In the set of six common letters in English, `e t o a i n`, consonants were less likely to be deleted than vowels. Surprisingly, the six least common letters, `z q x j v k` were often deleted. Considering letter position in words, 14% of first letters, 35% of last letters, and 90% of middle letters were deleted.

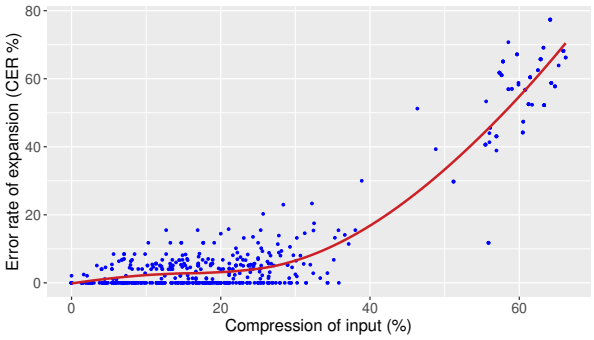

Figure 1: Error rate of automatic expansion with increasing abbreviation of the input.

Our study confirmed some of our initial beliefs about how people would do free-form abbreviation. We found people deleted vowels more frequently than consonants and people usually retained the first letter of words. Other aspects we found surprising such as the frequent deletion of uncommon letters. The percentage of middle letters deleted was high. One reason for this was some workers persistently only used the first letter of each word.

3.2 Initial Automatic Expansion Experiment

We selected 564 passages where each word was an in-order subset of the full word. We implemented a search that proposed inserting all characters at all positions in words in workers' input. The search was guided by the language models described in Vertanen et al. (2015). We used beam search to keep the search tractable. See the appendix for example input and the expanded output.

We measured accuracy using character error rate (CER). CER is the number of insertions, substitutions, and deletions required to transform the expanded text into the original text (typically multiplied by 100). As shown in Figure 1, the expansion had a CER of less than 5% for compression of up to 30%. Beyond that, much of the input was only the first letter of each word and our algorithm simply imagined probable text consistent with the provided letters. We think these results are promising given our search simply proposed the insertion of all characters at all positions.

4 Conversational Language Modeling

We think abbreviated input may most benefit users with slow input. From this point on, we focus on optimizing our system for use by Augmentative and Alternative Communication (AAC) users. AAC users may not be able to speak due to a condition

such as ALS. AAC users slow input rate make taking part in conversations difficult (Arnott et al., 1992). Sentence abbreviation may be particularly useful for short phrases with predictable language.

Our search-based approach to abbreviation expansion relies crucially on a well-trained language model. For a language model to work well it needs to be trained on data that is suited to the target domain. Ideally we would train our language models on large amounts of conversational communications written by AAC users. For privacy and ethical reasons, it is difficult to find large amounts of such data. Therefore, in this section, we explore selecting training data from an out-of-domain dataset using a small amount of in-domain AAC-like data.

4.1 Selecting Training Data

As our in-domain set, we used 29 K words of AAC-like crowdsourced messages (Vertanen and Kristensson, 2011b). For our out-of-domain training set, we used one billion words of web text from Common Crawl[1]. We only kept sentences consisting of A–Z, apostrophes, spaces, commas, periods, question marks, and exclamation point. We compared three ways to select training sentences:

Random selection. We randomly selected sentences until we reached 100 million characters.

Cross entropy difference selection. Following Moore and Lewis (2010), we trained an in-domain 4-gram word language model on our AAC-like data, and an out-of-domain 4-gram model on a random subset of web text (disjoint from the training set). We calculated the cross-entropy difference of training sentences using the in- and out-of-domain models. We selected the highest scoring sentences until we reached 100 million characters.

BERT selection. BERT is a language representation model built using self-attentive transformers (Devlin et al., 2019). We took the in- and out-of-domain data from the previous step and labeled each sentence based on its set. We then trained a binary classifier using `bert-base-uncased`[2]. We ran our classifier on each sentence in the training set yielding the probability of a sentence belonging to the in-domain set. We selected the top sentences until we reached 100 million characters.

4.2 Comparison of Selection Methods

As shown in Table 1, random sentences from Common Crawl averaged 30 words. The cross-entropy difference and BERT methods selected shorter sentences of 14 and 11 words respectively. This is likely good given our goal of supporting short, conversational messages. For comparison, sentences averaged 13 words in the in-domain AAC set and 10 words in DailyDialog (Li et al., 2017). DailyDialog consists of two-sided everyday dialogues.

Method	Words sent.	OOV (%)	Enron ppl	Daily ppl	Enron CER
Random	29.8	1.21	4.81	3.31	7.04
CE diff.	14.3	0.32	4.57	3.11	6.00
BERT	11.1	0.39	4.53	3.05	5.75

Table 1: Impact of selection method on training sentences and performance of letter language models.

We calculated the out-of-vocabulary (OOV) rate with respect to a vocabulary of 100 K words. Our randomly selected sentences had a much higher 1.2% OOV rate compared to cross-entropy and BERT selected data at 0.3% and 0.4% respectively (Table 1). Again this suits our purpose as we suspect abbreviated input is best suited for sentences without uncommon words. For comparison, the OOV rates of DailyDialog and our AAC-like set were both low at 0.2%. See the appendix for samples of sentences selected by each method.

We trained 12-gram character language models with Witten-Bell smoothing on each 100 million character training set. We trained without count cutoffs and did not prune the models. The binary BerkeleyLM (Pauls and Klein, 2011) size of the random, cross-entropy difference, and BERT models were 1.7 GB, 1.3 GB, and 1.2 GB respectively.

We evaluated these character language models on the Enron mobile (Vertanen and Kristensson, 2011a) and DailyDialog (Li et al., 2017) datasets. Before evaluation, we split each dialog turn in DailyDialog into single sentences and randomized their order. We calculated the average per-character *perplexity* of these two datasets. As shown in Table 1, the cross-entropy and BERT models had perplexities around 6% lower than the random model with the BERT model having the lowest perplexity.

We also compared the recognition accuracy of the three language models using the recognizer and data to be described in the next section. As shown in Table 1 (right column), these perplexities reductions did translate into improvements in recognition accuracy on touchscreen input where spaces and

[1] https://commoncrawl.org/

[2] https://github.com/google-research/bert/

50% of mid-word vowels were removed.

5 Recognizing Noisy Abbreviated Input

We now describe how we used our optimized language models to recognize noisy abbreviated input.

5.1 Decoder Details and Improvements

We extended the VelociTap touchscreen keyboard decoder (Vertanen et al., 2015). VelociTap searches for the most likely text given a sequence of 2D taps. Each tap has a likelihood under a 2D Gaussians centered at each key. Taps can be deleted without generating a character by incurring a deletion penalty. Adding characters to a hypothesis incur penalties based on a character language model.

The decoder can insert characters without consuming a tap. A general insertion penalty allows all possibles characters to be inserted. The decoder also has separate space and apostrophe insertion penalties. We extend this further by adding a *vowel insertion penalty* for inserting the vowels: `a`, `e`, `i`, `o`, `u`. However, this penalty is only used if the prior character is not a space. This models that vowels should not be skipped at the start of words.

The search is performed in parallel, with different threads extending partial hypotheses. When a hypothesis consumes all taps, it is added to an n-best list. To keep the search tractable, a configurable beam controls whether partial hypotheses are pruned. A wider beam searches more thoroughly, but at the cost of more time and memory.

To date, VelociTap has only used n-gram language models. We extend the decoder to use a recurrent neural network language model (RNNLM) either as a replacement for the character n-gram during search, or to rescore the n-best list. When used for rescoring, we compute the log probability of each sentence under the RNNLM. We multiply this probability by an *RNNLM scale factor* and add the result to a hypothesis' log probability.

We trained an RNNLM on the BERT-selected training data. After a hyperparameter search, we settled on 512 LSTM units, a character embedding size of 64, two hidden layers, a learning rate of 0.001, and a dropout probability of 0.5. We trained using the Adam optimizer. On the Enron Mobile and DailyDialog test sets, our RNNLM had a perplexity of 4.50 and 2.64 respectively.

To allow efficient hypothesis extension during RNNLM-based search, we augmented our partial hypotheses to track the state of the neural network. However, as we will see, RNNLM search required substantial memory and computation time. While we experimented with using a GPU for RNNLM queries, we found parallel CPU search was faster.

5.2 Touchscreen Data and Simulation Details

We tested our improvements on noisy, abbreviated, touchscreen keyboard input. We wanted noisy input to ensure our system was robust to mistakes AAC users may make when typing (e.g. when using a mouth stick or an eye-tracker). We created a test and development set using data collected on touchscreen phones (Vertanen et al., 2015, 2013) and watches (Vertanen et al., 2018, 2019). We limited our data to sentences from the Enron Mobile set. We concatenated taps to create single sentence sequences without spaces. We removed sentences where the number of taps did not match the length of its reference. This resulted in a test and development set of 6,631 and 731 sentences respectively.

We played back taps to our decoder, deleting mid-word vowels with a given *vowel drop probability*. We tested drop probabilities of 0.5 and 1.0. In our test set, 17.7% of characters were spaces. With a drop probability of 0.5, 27.9% of characters (including spaces) were deleted. If all mid-word vowels were dropped, 38.2% of characters were saved. For the n-gram search and RNNLM rescoring setups and two drop probabilities, we tuned decoder parameters to minimize CER on the development set. Tuning used a random restart hill-climbing approach. We tuned each of the four setups for 600 CPU hours. Due to the computational costs, we used the parameters found for the n-gram search for the RNNLM search.

We report the character error rate (CER), as well as word error rate (WER), and sentence error rate (SER) on our test set. We also report the Top-5 SER which is the lowest SER of the top five hypotheses. We searched in parallel using 24 threads on a dual Xeon E5-2697 v2 server. This large number of threads mainly sped up the RNNLM search.

5.3 Recognition Results

As shown in Table 2, using the RNNLM in the search instead of the n-gram model reduced error rates by 23% and 12% relative for a vowel drop probability of 0.5 and 1.0 respectively. This however came at a much higher cost with decoding taking much longer and requiring more memory. Using the n-gram model for search and rescoring with the RNNLM resulted in similar error rates

Decoder search	Drop prob.	CER (%)	WER (%)	SER (%)	Top-5 SER (%)	Decode time (s)	Memory (GB)
n-gram search	0.5	5.7 ± 0.3	12.4 ± 0.5	35.1 ± 1.1	22.0	0.21	40.7
+ RNNLM rescore	0.5	4.4 ± 0.2	9.5 ± 0.5	27.7 ± 1.1	16.5	0.34	52.1
RNNLM search	0.5	4.3 ± 0.2	9.3 ± 0.5	27.6 ± 1.1	15.4	24.05	353.2
n-gram search	1.0	9.5 ± 0.4	19.0 ± 0.7	45.5 ± 1.2	30.3	0.03	41.4
+ RNNLM rescore	1.0	8.0 ± 0.3	15.5 ± 0.6	38.5 ± 1.2	24.2	0.09	52.9
RNNLM search	1.0	8.2 ± 0.3	15.8 ± 0.6	41.5 ± 1.2	26.3	1.09	52.4

Table 2: Error rates and decoder performance using different search methods and vowel drop probabilities. ± values denote sentence-wise 95% bootstrap confidence intervals (Bisani and Ney, 2004).

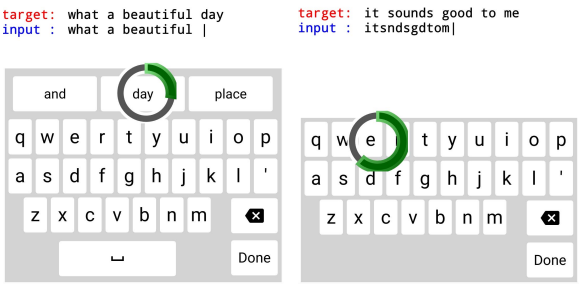

Figure 2: The word (left) and sentence (right) keyboard modes from our user study. The circle is centered on the user's touch location with a green arc showing progress towards the one second dwell time.

to searching with the RNNLM, but only caused modest increases in decode time and memory.

Dropping half of vowels, we recognized the correct sentence 72% of the time using RNNLM rescoring. If we assume an interface allowing selection from the top five results, this increased to 85%. Dropping all vowels was harder; we recognized the correct sentence only 59% of the time. Providing the top five sentences increased this to 74%.

Interestingly, our vowel drop probability 1.0 setups were faster. We investigated this by varying the tuned beams, measuring CER on the development set. We found for drop 0.5, a narrower beam increased CER while a wider beam provided no gain. For drop 1.0, a narrower beam also increased CER, but even a modestly wider beam increased CER slightly (3% relative). The tuned penalty for vowel insertion was small (0.8 probability). We observed in sentences with errors at a narrow beam, a wider beam sometimes resulted in more inserted vowels. This may have allowed more probable text, but ultimately a higher CER. This suggests we may need a more nuanced model of how users abbreviate, e.g. by penalizing contiguous vowel insertions.

6 User Study

Thus far, we tested abbreviated sentence input only in offline experiments. To see if our method offers competitive performance in practice, we conducted a user study using a touchscreen web application.

6.1 Design

We designed a touchscreen keyboard that runs in a mobile web browser. The keyboard has two modes:

Word — This mode has the keys A–Z, apostrophe, spacebar, and backspace (Figure 2, left). The keyboard has three prediction slots above the keyboard. The left slot shows the exact letters typed. The center and the right slots show predictions based on a user's taps and any previous text. Predictions and recognition occur after each key press. Pressing the spacebar normally selects the left slot. Similar to the iPhone keyboard, if a user's input is noisy and we predict an auto-correction with high probability, we highlight this slot instead. In this case, pressing spacebar selects the auto-correction. A done button signals completion of a sentence.

Sentence — This mode is similar but has no spacebar or suggestion slots (Figure 2, right). Input is recognized only after the done button is pressed.

To simulate users with a slow input rate, users had to dwell on a key for one second to click it. We chose one second because this is a common default setting in dwell-based eye typing, for example, 1.2 seconds in Tobii Communicator. We display a progress circle around a user's finger location showing the dwell time. After a click, the keyboard border flashes and the nearest key is added to the text area above the keyboard.

Due to memory and computation requirements, we ran our decoder on a server at our university. The keyboard client makes requests to the server to recognize input. In word mode, at the start of

Metric	WORD	SENTENCE	Statistical test
Entry rate (wpm)	9.9 ± 1.5 [6.6, 12.4]	9.0 ± 1.5 [5.7, 11.5]	t(27) = -3.92, r = 0.60, $p < 0.001$
Error rate (CER %)	0.3 ± 0.5 [0.0, 2.5]	7.2 ± 5.4 [1.0, 23.6]	t(27) = 6.72, r = 0.79, $p < 0.001$

Table 3: User performance in each condition in our user study. Results formatted as: mean ± SD [min, max].

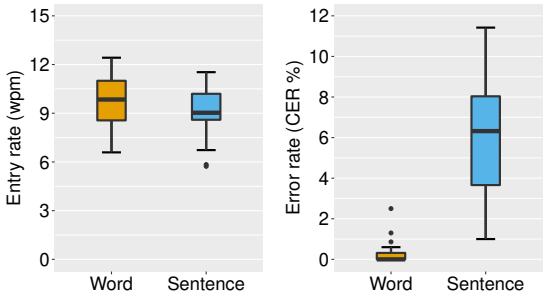

Figure 3: Entry and error rate in our user study.

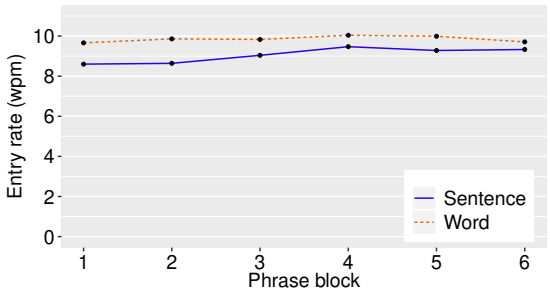

Figure 4: Entry rates for each block of four phrases.

each key press, we request predictions for the keyboard slots. In sentence mode, we request sentence recognition at the start of pressing the done button. By making the server request at the start of a key press, we effectively eliminated the need to wait for predictions. The average round trip time for requests in our user study was 0.41 s (sd 0.21) in the word mode and 0.58 s (sd 0.29) in sentence mode.

6.2 Procedure

We recruited 28 Amazon Mechanical Turk workers. The study took 30–40 minutes. Workers were paid $10. We also offered a $5 bonus for the fastest 10% of workers in each condition subject to having a CER below 5%. This was a within-subject experiment with two counterbalanced conditions: WORD and SENTENCE. The conditions used the word and sentence mode of the keyboard respectively.

Workers typed 26 phrases in each condition. The first two were practice phrases which we did not analyze. Workers wrote phrases written by people with ALS for voice banking purposes (Costello, 2014). We used phrases with 3–6 words (1,182 total phrases). Workers received a random set of phrases and never wrote the same phrase twice.

6.3 Results

Table 3 and Figure 3 show results and statistical tests. We calculated *entry rate* in words-per-minute (wpm). We considered a word to be five characters including space. We measured the entry time from a worker's first tap until they finished dwelling on the done button. The entry rate in WORD was faster at 9.9 wpm versus SENTENCE at 9.0 wpm. This difference was significant (Table 3).

As shown in Figure 4, participants started out slower in SENTENCE compared to WORD, but the entry rate gap closed as they wrote more phrases. We averaged performance in the first eight and last eight phrases. In WORD, the entry rate was 9.7 wpm in the first set and 9.9 wpm in the last set. In SENTENCE, the entry rate was 8.6 wpm in the first set and 9.6 wpm in the last set. This is promising, as perhaps with more practice, sentence abbreviation might achieve comparable speed but without requiring monitoring of word predictions.

Participants were less accurate in SENTENCE with a CER of 7.2% versus 0.3% in WORD. This difference was significant (Table 3). Participants obtained a completely correct phrase 97% of the time in WORD, but only 68% in SENTENCE. We think the lower accuracy in SENTENCE was mostly due to some users abbreviating phrases too aggressively. In phrases recognized completely correctly, the compression rate was 35%. In phrases with recognition errors, the compression rate was 43%.

We classified phrases in SENTENCE according to their input length versus the reference length minus spaces and mid-word vowels. 252 phrases had the correct length, 162 were longer, and 258 were shorter. These sets correspond to phrases that were likely correctly abbreviated, under-abbreviated, and over-abbreviated. The error rates of these sets were 3.2%, 2.1%, and 14.0% respectively. We found five workers over-abbreviated 20 or more phrases. Removing these workers lowered the overall CER to 5.7%. While not as accurate as word input, sentence input did have acceptable accuracy

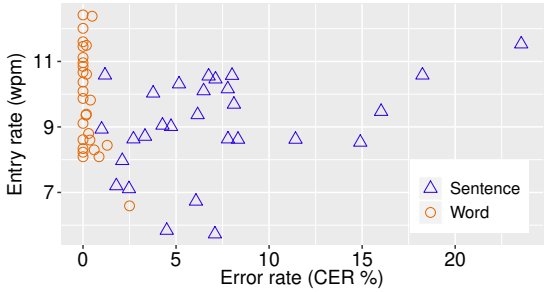

Figure 5: Participants' entry and error rate in each condition of the user study.

when users abbreviated as instructed.

Individual user performance was variable (Figure 5). 16 participants achieved 0% CER in WORD and all but two had a CER below 1%. While in SENTENCE, no participant achieved 0% CER and five participants had a high CER of over 10%.

Using backspace, participants could fix incorrect letters or misrecognized words. The number of backspaces per final output character was low at 0.02 in both conditions. Thus, it appears participants precisely targeted keys, likely as a result of the slow input induced by the dwell time.

7 Discussion

We set out to show we could accelerated the writing of short and reasonably predictable phrases by combining sentence-at-a-time recognition with aggressive abbreviation. In our final user study, we found our method did not quite beat a conventional keyboard with word predictions. However, users in our study likely had substantial experience doing word-at-a-time input on their phones. It appears users got faster at abbreviated sentence input even during the brief study session. By the last eight phrases, users were only 3% relative slower using sentence abbreviated input compared to word-at-a-time input with word completions. When users provided abbreviated input consisting of all the correct letters except mid-word vowels, 90% of these phrases were expanded correctly.

We observed the abbreviation behaviour of a large number of non-AAC users and designed a system supporting the most common behaviours. While we could have tried to learn abbreviation behaviors from actual AAC user data, this presents a number of issues. First, actual text from AAC users is difficult to obtain for ethical and privacy reasons. While it may be possible to obtain such text via donations from AAC users and from online sources, such sources lack visibility into how the user actually produced the text (e.g. did they use word completions?). Further, the propensity to abbreviate may be influenced by the particular AAC interface used. Second, even if we could source AAC abbreviated text, we would have no reliable way to determine the unabbreviated text. We could have asked AAC users to complete our abbreviation study, but this would have introduced more noise (incorrect key presses) that would have complicated our first study's goal of discovering natural abbreviation behaviors. It would have also limited the number of people we learned behaviors from. In this phase our goal was to discover what letters humans think are the most information carrying in a passage of text. While we suspect abbreviation strategies of AAC users would be similar, this would benefit from validation with AAC users.

We tested our method on touchscreen data recorded in previous studies on phones and watches, and in a web-based crowdsourced user study. We think our method mainly would benefit users who have a slow input rate; fast typists may only be slowed by the cognitive overheads of deciding what letters to omit or by disrupting their muscle memory for typing familiar words. This led us to limiting the input rate in our study by requiring users dwell for one second. While this study allowed us to confirm our abbreviation method is competitive with a conventional keyboard with word predictions, this needs validation with users with actual input rate limits. AAC user interaction may feature more imprecise key presses, more accidental key presses, and may introduce complications related to attending to word predictions (e.g. the "midas touch" problem in eye tracking). Further, we only tested one input rate, it is possible our method may be better or worse at different input speeds. We think our approach may also offer advantages for eyes-free text input, but this also needs comparison against conventional eyes-free input approaches (e.g. iPhone's VoiceOver feature).

We investigated abbreviation by omitting mid-word vowels. We did not investigate other forms of abbreviation such as phonetic replacement (e.g. "you" $\rightarrow$ "u") or removal of consonants. Our model may benefit from more sophisticated modeling on how and when vowels are inserted (e.g. penalizing repeated vowel insertions). Ideally improved models would be based on data collected by users engaged in actual abbreviated input. As our

results show, correctly inferring the intended sentences was challenging even when we asked users to obey a few simple behaviours, namely removing spaces and mid-word vowels. While an ideal system would support a wide-range of abbreviation behaviors and even adapt to individuals, we suspect this may be challenging given our current lack of training data on this task.

In our initial study, participants abbreviated email text that was displayed visually. An alternative approach would be to play audio of the text. While this might be a more realistic abbreviation task, it also presents practical challenges to participants such as remembering the text and spelling any difficult words. Perhaps an even more externally valid approach would be to have workers compose novel abbreviated sentences. This would require another step to obtain the unabbreviated compositions (Vertanen and Kristensson, 2014; Gaines et al., 2021). Given we now have a competent initial system, it would be interesting to undertake such a data collection effort.

Our results suggest a simple correction interface based on selecting from the top sentences would often, but not always work. Designing an efficient and easy-to-use interface for correcting a few words within such sentence results would be interesting future work. This might be especially challenging to design for users with diverse motor abilities.

We used language models trained on only 100 M characters of text. While this allowed us to compare the efficacy of the language model types and decoder configurations, substantially more training data is available along with neural architectures that scale to large training sets, e.g. GPT-2 (Radford et al., 2019). We suspect further recognition accuracy gains are possible for abbreviated, noisy input by incorporating such models. Further, we could likely obtain additional improvements from the n-gram model by training on more data and then pruning the model to reduce its size. We avoided doing this in this work to fairly compare the n-gram and RNN language models when trained on the same amount of text.

Our language model training data was drawn from Common Crawl. We used a corpus of AAC-like crowdsourced messages to select training sentences from Common Crawl. Other sources of training data such as Twitter or Reddit are likely more conversational in style. It would be interesting to investigate whether data selecting from a more targeted large-scale training source provides additional improvements in language modeling.

We did not specifically investigate how our method would support text containing difficult words such as acronyms or proper names. Users can often anticipate and alter their input behavior to avoid auto-correct errors, e.g. by force (Weir et al., 2014), by long pressing a key (Vertanen et al., 2019), or by switching to a precise input mode (Dudley et al., 2018). Similarly, our abbreviated input method needs a way to specify words that should not be expanded or auto-corrected.

At the onset, we did not know that our proposed abbreviation technique would be competitive to conventional word completion. The results from our user study tell us we need to make further improvements to our recognition, better train users to abbreviate in supported ways, and conduct a longitudinal evaluation. Further, testing an abbreviated input prototype with AAC users will undoubtedly lead to new insights. This paper is a first step in producing a viable prototype for testing with users with rate-limited input abilities.

8 Conclusion

We explored accelerating text communication by abbreviated sentence input. We conducted a user study to learn how users abbreviate. We showed the efficacy of a neural classifier to select conversational-style training instances from a large text corpus. We found that dropping spaces and mid-word vowels can provide compression of sentences from 28% to 38%. Such abbreviated and noisy input can often be expanded correctly 59% to 72% of the time. We also showed how the accuracy of a statistical virtual keyboard decoder can be improved by using a neural language model to re-rank the top recognition results. Finally, after practice, users wrote only slightly slower using sentence abbreviated input at 9.6 words-per-minute compared to a conventional keyboard with word predictions at 9.9 words-per-minute. If a phrase was abbreviated by removing spaces and mid-word vowels, our system expanded the abbreviated input to the intended phrase 90% of the time.

Acknowledgments

This material is based upon work supported by the NSF under Grant No. IIS-1750193.

References

John L. Arnott, Alan F. Newell, and Norman Alm. 1992. Prediction and Conversational Momentum in an Augmentative Communication System. *Communications of the ACM*, 35(5):46–57.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain Adaptation via Pseudo In-Domain Data Selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

M. Bisani and H. Ney. 2004. Bootstrap Estimates for Confidence Intervals in ASR Performance Evaluation. *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 409–411.

Boxing Chen and Fei Huang. 2016. Semi-supervised Convolutional Networks for Translation Adaptation with Tiny Amount of In-domain Data. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 314–323, Berlin, Germany. Association for Computational Linguistics.

Boxing Chen, Roland Kuhn, George Foster, Colin Cherry, and Fei Huang. 2016. Bilingual Methods for Adaptive Training Data Selection for Machine Translation. In *Proceedings of the Association for Machine Translation in the Americas*, pages 93–103.

Mara Chinea-Rios, Germán Sanchis-Trilles, and Francisco Casacuberta. 2018. Creating the Best Development Corpus for Statistical Machine Translation Systems. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, pages 99–108. European Association for Machine Translation.

John M. Costello. 2014. Message banking, voice banking and legacy messages. *Boston Children's Hospital, Boston, MA*.

Patrick W. Demasco and Kathleen F. McCoy. 1992. Generating Text from Compressed Input: An Intelligent Interface for People with Severe Motor Impairments. *Communications of the ACM*, 35(5):68–78.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

John J. Dudley, Keith Vertanen, and Per Ola Kristensson. 2018. Fast and precise touch-based text entry for head-mounted augmented reality with variable occlusion. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 25(6).

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation Data Selection using Neural Language Models: Experiments in Machine Translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 678–683, Sofia, Bulgaria. Association for Computational Linguistics.

Dylan Gaines, Per Ola Kristensson, and Keith Vertanen. 2021. Enhancing the composition task in text entry studies: Eliciting difficult text and improving error rate calculation. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA. Association for Computing Machinery.

Jianfeng Gao, Joshua Goodman, Mingjing Li, and Kai-Fu Lee. 2002. Toward a Unified Approach to Statistical Language Modeling for Chinese. *ACM Transactions on Asian Language Information Processing*, 1(1):3–33.

Guillem Gascó, Martha-Alicia Rocha, Germán Sanchis-Trilles, Jesús Andrés-Ferrer, and Francisco Casacuberta. 2012. Does More Data Always Yield Better Translations? In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, page 152–161, USA. Association for Computational Linguistics.

Shaona Ghosh and Per Ola Kristensson. 2017. Neural Networks for Text Correction and Completion in Keyboard Decoding. *arXiv preprint arXiv:1709.06429*.

Ellyn Gregory, Melinda Soderman, Christy Ward, David R Beukelman, and Karen Hux. 2006. AAC Menu Interface: Effectiveness of Active versus Passive Learning to Master Abbreviation-Expansion Codes. *Augmentative and Alternative Communication*, 22(2):77–84.

Sangmok Han, David R. Wallace, and Robert C. Miller. 2009. Code Completion from Abbreviated Input. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, ASE '09, page 332–343, USA. IEEE Computer Society.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the Translation Model for Statistical Machine Translation Based on Information Retrieval. In *Proceedings of the 10th EAMT Conference: Practical applications of machine translation*, pages 133–142, Budapest, Hungary. European Association for Machine Translation.

Brian Hur, Timothy Baldwin, Karin Verspoor, Laura Hardefeldt, and James Gilkerson. 2020. Domain Adaptation and Instance Selection for Disease Syndrome Classification over Veterinary Clinical Notes. In *Proceedings of the 19th SIGBioMed Workshop on*

Biomedical Language Processing, pages 156–166, Online. Association for Computational Linguistics.

Per Ola Kristensson and Shumin Zhai. 2004. SHARK^2: A Large Vocabulary Shorthand Writing System for Pen-based Computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, pages 43–52, New York, NY, USA. ACM.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Sung-Chien Lin, Chi-Lung Tsai, Lee-Feng Chien, Ker-Jiann Chen, and Lin-Shan Lee. 1997. Chinese Language Model Adaptation Based on Document Classification and Multiple Domain-Specific Language Models. In *Proceedings of European Conference on Speech Communication and Technology*, pages 1463–1466.

Yajuan Lü, Jin Huang, and Qun Liu. 2007. Improving Statistical Machine Translation Performance by Training Data Selection and Optimization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 343–350, Prague, Czech Republic. Association for Computational Linguistics.

Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Domain Adaptation with BERT-based Domain Classification and Data Selection. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 76–83, Hong Kong, China. Association for Computational Linguistics.

Saab Mansour, Joern Wuebker, and Hermann Ney. 2011. Combining Translation and Language Model Scoring for Domain-Specific Data Filtering. In *International Workshop on Spoken Language Translation (IWSLT) 2011*.

Robert C. Moore and William Lewis. 2010. Intelligent Selection of Language Model Training Data. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 220–224, Stroudsburg, PA, USA. Association for Computational Linguistics.

Emli-Mari Nel, Per Ola Kristensson, and David J. C. MacKay. 2019. Ticker: An Adaptive Single-Switch Text Entry Method for Visually Impaired Users. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(11):2756–2769.

Hugo Nicolau, André Rodrigues, André Santos, Tiago Guerreiro, Kyle Montague, and João Guerreiro. 2019. The Design Space of Nonvisual Word Completion. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '19, page 249–261, New York, NY, USA. Association for Computing Machinery.

Zuzanna Parcheta, Germán Sanchis-Trilles, and Francisco Casacuberta. 2018. Data Selection for NMT using Infrequent n-gram Recovery. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, pages 219–227. European Association for Machine Translation.

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: Human Language Technologies - Volume 1*, HLT '11, pages 258–267, Stroudsburg, PA, USA. Association for Computational Linguistics.

Álvaro Peris, Mara Chinea-Ríos, and Francisco Casacuberta. 2017. Neural Networks Classifier for Data Selection in Statistical Machine Translation. *The Prague Bulletin of Mathematical Linguistics*, 108(1):283–294.

Stefano Pini, Sangmok Han, and David R. Wallace. 2010. Text Entry for Mobile Devices Using Ad-Hoc Abbreviation. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, page 181–188, New York, NY, USA. Association for Computing Machinery.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models Are Unsupervised Multitask Learners. *OpenAI blog*, 1(8):9.

Anthony Rousseau. 2013. XenC: An Open-Source Tool for Data Selection in Natural Language Processing. *The Prague Bulletin of Mathematical Linguistics*, 100:73–82.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19, Montréal, Canada. Association for Computational Linguistics.

Stuart M Shieber and Rani Nelken. 2007. Abbreviated Text Input using Language Modeling. *Natural Language Engineering*, 13(2):165–183.

Kumiko Tanaka-Ishii, Yusuke Inutsuka, and Masato Takeichi. 2001. Japanese Text Input System With Digits. In *Proceedings of the First International Conference on Human Language Technology Research*.

Keith Trnka, John McCaw, Debra Yarrington, Kathleen F. McCoy, and Christopher Pennington. 2009.

User Interaction with Word Prediction: The Effects of Prediction Quality. *ACM Transactions on Accessible Computing*, 1:17:1–17:34.

Keith Vertanen, Crystal Fletcher, Dylan Gaines, Jacob Gould, and Per Ola Kristensson. 2018. The Impact of Word, Multiple Word, and Sentence Input on Virtual Keyboard Decoding Performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '18, pages 626:1–626:12, New York, NY, USA. ACM.

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 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–14, New York, NY, USA. Association for Computing Machinery.

Keith Vertanen and Per Ola Kristensson. 2011a. A Versatile Dataset for Text Entry Evaluations Based on Genuine Mobile Emails. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices & Services*, MobileHCI '11, pages 295–298, New York, NY, USA. ACM.

Keith Vertanen and Per Ola Kristensson. 2011b. 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*, pages 700–711, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Keith Vertanen and Per Ola Kristensson. 2014. Complementing text entry evaluations with a composition task. *ACM Transactions of Computer Human Interaction*, 21(2):8:1–8:33.

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 SIGCHI Conference on Human Factors in Computing Systems*, CHI '15, pages 659–668, New York, NY, USA. ACM.

Keith Vertanen, Haythem Memmi, and Per Ola Kristensson. 2013. The Feasibility of Eyes-free Touchscreen Keyboard Typing. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '13, pages 69:1–69:2, New York, NY, USA. ACM.

Daryl Weir, Henning Pohl, Simon Rogers, Keith Vertanen, and Per Ola Kristensson. 2014. Uncertain Text Entry on Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 2307–2316, New York, NY, USA. ACM.

Tim Willis, Helen Pain, and Shari Trewin. 2005. A Probabilistic Flexible Abbreviation Expansion System for Users with Motor Disabilities. In *Proceedings of the 2005 International Conference on Accessible Design in the Digital World*, Accessible Design'05, page 4, Swindon, GBR. BCS Learning & Development Ltd.

Tim Willis, Helen Pain, Shari Trewin, and Stephen Clark. 2002. Informing Flexible Abbreviation Expansion for Users with Motor Disabilities. In *Proceedings of the 8th International Conference on Computers Helping People with Special Needs*, ICCHP '02, page 251–258, Berlin, Heidelberg. Springer-Verlag.

Keiji Yasuda, Ruiqiang Zhang, Hirofumi Yamamoto, and Eiichiro Sumita. 2008. Method of Selecting Training Data to Build a Compact and Efficient Translation Model. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Shumin Zhai and Per Ola Kristensson. 2008. Interlaced QWERTY: Accommodating Ease of Visual Search and Input Flexibility in Shape Writing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 593–596, New York, NY, USA. Association for Computing Machinery.

Shumin Zhai, Alison Sue, and Johnny Accot. 2002. Movement Model, Hits Distribution and Learning in Virtual Keyboarding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, page 17–24, New York, NY, USA. Association for Computing Machinery.

Suwen Zhu, Tianyao Luo, Xiaojun Bi, and Shumin Zhai. 2018. Typing on an Invisible Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '18, pages 439:1–439:13, New York, NY, USA. ACM.

Appendix

A Abbreviation Study Instructions

In our first crowdsourced study, we had 200 workers abbreviate a series of ten email messages. Figure 6 shows the complete instructions we gave to workers.

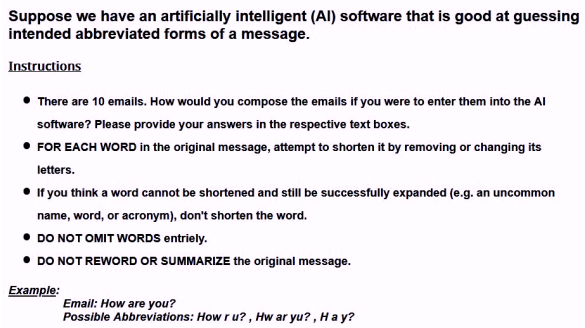

Figure 6: Instructions given to workers in our crowdsourced free-from abbreviation study.

B Error Rate to Compression

In our final crowdsourced study, we plotted participants' character error rate against increasing abbreviation in the SENTENCE condition. With increasing compression, the error rate also increased (Figure 7).

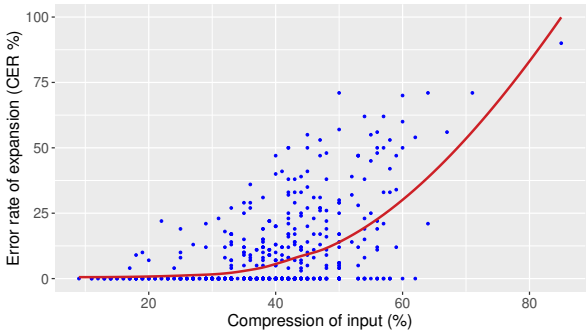

Figure 7: Error rate of automatic expansion with increasing abbreviation of the input in the final study.

C Initial Automatic Expansion Experiment

Table 4 shows some examples from our initial automatic expansion experiment where the decoder inserted all characters at all possible positions in a worker's input.

D Selecting Training Data

Table 5 shows a list of example sentences selected using three ways: random selection, cross-entropy difference selection, and BERT selection.

E Recognizing Noisy Abbreviated Input

Table 6 shows some examples of recognition results using the RNNLM rescoring configuration. A complete list of recognition results is provided in our supplementary materials.

Original	we are off to the uk in a couple of days
Abbreviation	w r off t th uk n a cpl of dys
Expansion	we are off to the uk in a couple of days
Original	didn't get a commitment just told them i thought it would be impossible
Abbreviation	dnt gt a cmmt jst tld thm i tht it wd b mpss
Expansion	don't get a comment just told them i thought it would be impress
Original	no arrangements he just hasn't had a good year on a comparative basis
Abbreviation	n ats he j h h a go ye on a ct b
Expansion	and that's the job he has a good eye on a city but

Table 4: Examples from the initial automatic expansion experiment in Section 3.2. Each example shows the original text, workers' abbreviated version, and the automatically expanded result. Errors are underlined.

Random selection	
Random 1:	i'm a huge fan of your work it's really well done.
Random 2:	the main challenge is to integrate more and more qubits to silicon chips.
Cross-entropy difference selection	
Top 1:	what's for dinner? what's for lunch?
Top 2:	how's things going?
Mid 1:	want to know what happened during fish robert ed fish's life?
Mid 2:	do you want to work at a job or do you want to play at a passion.
Bottom 1:	think super mario bros.
Bottom 2:	is there a form applicants should submit, or should they just send an email with their resume?
BERT selection	
Top 1:	you don't like doing homework?
Top 2:	you need money?
Mid 1:	i am very proud of you for what you are doing with your life right now.
Mid 2:	imagine my terrible position!
Bottom 1:	she has a bachelor's degree in political science from lincoln university in oxford, pennsylvania.
Bottom 2:	good relations are answers. bad relations are disasters.

Table 5: Examples of selected text data using three different approaches in Section 4.1. For BERT and cross-entropy difference selection Top, Mid, and Bottom represent the absolute positions in the ordered list according to their scores.

Vowel drop probability	Example	
0.5	Reference:	hopefully this can wait until monday
0.5	Input:	hopefllltthisvamwwtujrlmndy
0.5	Recognition:	hopefully this can wait until monday
0.5	Reference:	let it rip
0.5	Input:	ltutrp
0.5	Recognition:	let it rip
0.5	Reference:	should systems manage the migration
0.5	Input:	shldsystensmnferhemgratin
0.5	Recognition:	she'd systems manager he migration
1.0	Reference:	could you see where this stands
1.0	Input:	cldyusewhtwrgsstnda
1.0	Recognition:	could you see where the stands
1.0	Reference:	florida is great
1.0	Input:	flrdaushrt
1.0	Recognition:	florida is great
1.0	Reference:	they are more efficiently pooled
1.0	Input:	yhyare'rrefgvmyluplf
1.0	Recognition:	they are more egg vinyl hold

Table 6: Examples of abbreviated and noisy input and the resulting recognition results for two different vowel drop probabilities in Section 5.3. The input text represents the closest key to each tap observation in our data. Recognition errors are underlined.