Just Speak It: Minimize Cognitive Load for Eyes-Free Text Editing with a Smart Voice Assistant

Jiayue Fan*
Chenning Xu*
Chun Yu†
Yuanchun Shi
fanjy18@tsinghua.org.cn
xcn18@tsinghua.org.cn
chunyu@tsinghua.edu.cn
shiyc@tsinghua.edu.cn
Department of Computer Science and Technology, Tsinghua University
Key Laboratory of Pervasive Computing, Ministry of Education
Beijing Academy of Artificial Intelligence
Beijing, China

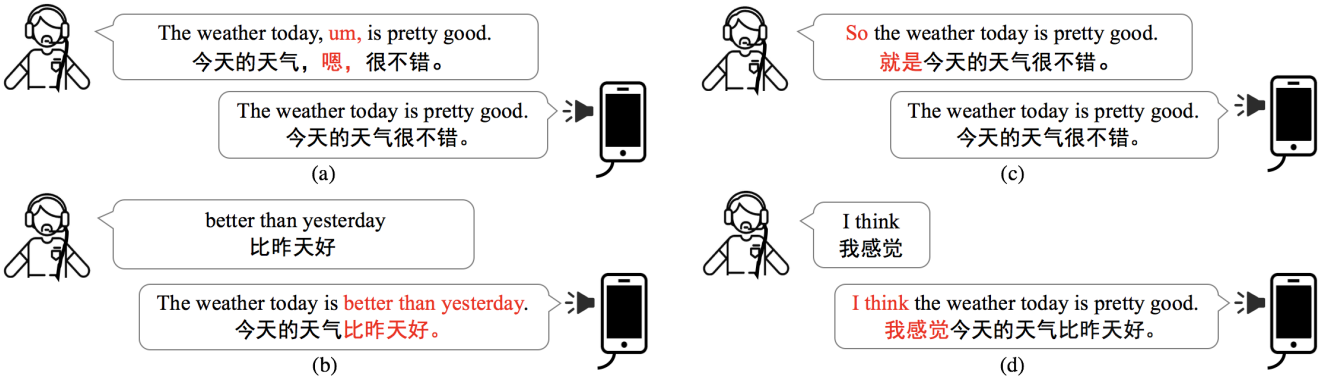


Figure 1: SmartEdit applies two strategies: (a) (c) removes colloquial inserts automatically after the users' speech; (b) (d) allows users to just speak the target words to perform insertion or replacement operations, and then SmartEdit will predict users' editing intention accordingly.

ABSTRACT

Entering text precisely by voice, users might encounter colloquial inserts, inappropriate wording, and recognition errors, which brings difficulties to voice editing. Users need to locate the errors and then correct them. In eyes-free scenarios, this select-modify mode brings a cognitive burden and a risk of error. This paper introduces neural networks and pre-trained models to understand users' revision intention based on semantics, reducing the need for the information from users' statements. We present two strategies. One is to remove the colloquial inserts automatically. The other is to allow users to edit by just speaking out the target words without having to say the context and the incorrect text. Accordingly, our approach can predict whether to insert or replace, the incorrect text to replace, and the position to insert. We implement these strategies in SmartEdit, an eyes-free voice input agent controlled with earphone buttons. The evaluation shows that our techniques reduce the cognitive load and decrease the average failure rate by 54.1% compared to descriptive command or re-speaking.

CCS CONCEPTS

• **Human-centered computing** → **Text input**.

KEYWORDS

Text editing, eyes-free, voice-based text editing, voice user interfaces, natural language processing.

*Both authors contributed equally to this research.
†This is the corresponding author.

UIST '21, October 10–14, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8635-7/21/10…$15.00
https://doi.org/10.1145/3472749.3474795

ACM Reference Format:
Jiayue Fan, Chenning Xu, Chun Yu, and Yuanchun Shi. 2021. Just Speak It: Minimize Cognitive Load for Eyes-Free Text Editing with a Smart Voice Assistant. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21), October 10–14, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3472749.3474795

1 INTRODUCTION

Voice-based text input is a natural and efficient input modality in eyes-free scenarios [32]. Using a voice assistant, users can reply to emails, write short messages, record notes, or compose drafts while their eyes and bodies are engaged in other tasks, such as walking and driving. This input modality is also very effective for visually impaired users [1]. Editing is an essential step for voice input. Users need to remove colloquial words, correct recognition errors, and revise inappropriate words to write precise paragraphs.

The linear [22] and temporal [16] nature of audio increases the mental burden and the error rate of voice editing [9]. Recent works proposed two strategies for voice editing. The first one was to use descriptive commands like *"Change bat to cat"*. EDITalk [9] adopted this strategy for core editing operations, including insertions, replacements, and deletions. However, users had to recall the original incorrect text and command syntax for the intended operation, which brought some cognitive burden [11]. The second one was to re-speak a portion of the original text that contained the wrong words (e.g., correcting *"the bat sat"* by saying *"the cat sat"*) [17, 29]. However, this strategy required the user to add sufficient context; otherwise, the editing might fail due to alignment error [28]. Users were uncertain about how much context to add, and probably attempted multiple times to complete a task [10].

We explore how to leverage Natural Language Processing (NLP) techniques to minimize the cognitive load of voice editing from two perspectives: automatic editing and assisted editing. We propose two strategies. The first one is to remove the colloquial words automatically. In a Wizard-of-Oz study, we found that removing meaningless colloquial insertions was a frequent operation and decreased the efficiency of voice input. Recent studies [7, 14, 33] on automatically removing colloquial inserts only considered filled pauses (e.g., *"uh"*, *"well"*). However, filled pauses were only a part of colloquial inserts. We observed other colloquial inserts in our study and developed a semantic-based algorithm to remove various colloquial inserts. The second one is to assist users to insert or replace a word (or a phrase) by just speaking the target word (or phrase). The target words may only contain changed words (e.g., speak *"The color"* to change *"He painted the nuts into the shape of cheese."* to *"He painted the nuts the color of cheese."*), or contain changed words and some context. Accordingly, our algorithm can predict the type of operation (insertion or replacement), the incorrect text for replacement, and the position for insertion. The prediction bases on how well the target word (or phrase) match the semantics of the original text, and how well it articulates with the context in different positions. These strategies reduce the demand for recalling original text and organizing complicated or repetitive commands and thus make revisions easier for users.

To implement these intelligent algorithms, we use the linguistic knowledge embedded in the neural networks within pre-trained language models. This knowledge enables our system to identify meaningless insertions and understand where the user wants to put the target words without relying on context or incorrect words given by users. We develop SmartEdit, an eyes-free smart voice input assistant on a smartphone that applies the two strategies. It adopts the buttons of a wired headset as the controller and allows users to input verbally, review by listening to the recognized text and revise the incorrect words. We conducted a user study to evaluate the usability of SmartEdit. Results showed that filtering colloquial inserts could reduce 27.3% editing tasks. Revising by just speaking out target words had a lower cognitive load than revising by descriptive commands or re-speaking, which decreased the average length of commands by 34.6% and reduced the average failure rate by 54.1%. The effectiveness of the two strategies demonstrates the feasibility of reducing editing efforts for voice input by leveraging the power of language models.

2 RELATED WORK

There are four broad areas related to our work: eyes-free word processing systems, the correction during voice input, Natural Language Processing(NLP) algorithms for text editing, and disfluency detection.

2.1 Eyes-Free Voice-based Input

Dictation was nearly five times as fast as keyboard-based text entry [1]. A complete voice input system needs to combine text dictation, text review, and recognized text correction in a closed-loop process.

Voice-based editing without visual feedback was first introduced by Ira A. Gerson [8] in a voice dialing system. Users could speak the telephone number chunk by chunk (e.g., 1-2-3; 4-5; 6-7), separated by the user-defined placement of pauses. After each pause, the system would read the recognized numbers of the last chunk. If any error occurred, users could use the command *"clear"* to delete the last chunk and re-speak it. Jan Cuřín [5] proposed an in-car text editor which adopted similar ideas. The editor allowed the user to speak short segments, review the text, and use a steering wheel interface to choose the incorrect segments or words to edit. However, it was hard and inefficient to re-speak the whole segment or choose the wrong word using word-by-word traversal.

Users could use voice commands supported by voice-based dictation applications like Dragon NaturallySpeaking[1] to control the entire process. For eyes-free scenarios, Ghosh et al. [9] designed a voice-only closed-loop text processing system. To perform core operations like *insert*, *replace* and *delete*, users could first demonstrate *where* and *how much* of the text needed change [2] and then dictate the target words (e.g., *"Insert cat before sat"*). Besides, the system also supported navigation operations to *repeat* and *restart* and meta operations to *comment on* and *highlight*. The cognitive load to remember the incorrect text and the effort to speak long commands was a challenge in EDITalk. Moreover, Halverson et al. [12] had proposed that *cascades* of mistakes reduced the efficiency of editing when the command was used or recognized incorrectly.

2.2 Editing in Verbal Input

In the voice input process, editing took more than 60% of the time [1] and placed loads of burdens on users. McNair and Weibel [17]

[1] https://www.nuance.com/dragon.html

had first proposed to complete selecting and modifying in one step. Users only needed to re-speak a phrase containing the modified words, e.g., correcting *"the bat sat"* by saying *"the cat sat"*. The system could find the corresponding incorrect text by aligning the re-speaking phrase with the original text. Much of previous research about the text alignment has a basis on phonetic similarity. McNair and Weibel [17] proposed to build a Finite State Grammar (FSG) based on the original text. The re-speaking phrase would run through a recognizer within this FSG and produced a subpiece of the original text, which was the text to be replaced. Vertanen and Kristensson [28] optimized this model by integrating some new elements into FSG, including the alternatives for each word, the unknown words, and the empty words. Other researchers also utilized Levenshtein distance [3] or phonetic similarity of the words [26] to align the text. However, these studies often relied on the context given by users to improve accuracy, unless the pronunciations of incorrect words were very similar to the right words. This restriction made users more prone to errors. Ghosh et al. [11] presented an eyes-free voice-based interaction that allowed users to use descriptive commands and re-speaking alternatively, though the shortcomings of these methods were unsolved. They also proposed a new re-speaking technique that utilized the stem, the lemma, the WordNet domain [18], and the linear interpolation [34] of bigrams and trigrams to align the re-speaking text with the original sentence. It did not make ample use of semantic information; hence the user was still required to provide the context in many cases.

Introducing other modalities was a feasible solution because it reduced the difficulty and uncertainty during incorrect text identification. Suhm et al. [27] had presented a multimodal error correction method that allowed users to correct recognition errors efficiently with speech, mouse, keyboard, and stylus. Korok Sengupta et al. [23] leveraged the user's point of gaze to select the incorrect word. However, these interfaces only work if users sit in front of a desktop or personal computer. Debjyoti Ghosh et al. [10] implemented EYEditor, a heads-up smartglass-based text editor for on-the-go users. It allowed users to navigate through sentences, select text, and undo/redo with a manual input interface. For editing, EYEditor supports both re-speaking and using the manual device to select the incorrect words. However, both methods had individual obstacles that led to inefficiency.

2.3 NLP Algorithms for Text Editing

Spelling error correction is a relevant problem with our task, which focuses on the automatic correction of wrong sentence. Wang et al. [30] proposed a Sequence-to-Sequence model with a confusion set to transform an input sentence into an error-free sentence. Zhang et al. [36] proposed a soft-masked transformer model to predict the true word at the wrong position. However, these automatic methods did not take users' editing intentions into account. If the predicted result was wrong, users could not intervene.

In keyboard-based input, users needed to provide the right words to correct the wrong sentence. Zhang et al. [35], presented three different text correction techniques to facilitate tap-based text keyboard typing. Two of the techniques, named $Drag-n-Throw$ and $Magickey$, utilized an RNN-based sequence-to-sequence model to target and error word according to the right one. Though, it relied on the spelling similarity and only supported the correction of one word.

2.4 Disfluency Detection

Non-fluent words are prevalent in spoken language. Users often pause during dictation since they tend to think about the previous expressions' correctness or conceive the following expressions. They may unconsciously insert some words during the pause with little contextual relevance [7], such as *"uh"*, *"I mean"*, and others. These words were called filled pauses. Many existing studies have explored the detection of disfluency and presented models such as sequence labeling model [14, 33], encoder-decoder model [31], and parsing-based model [21]. However, filled pause is only one category of non-fluent words. Additionally, the detection of more complex colloquial inserts can refer to these existing approaches.

3 STUDY 1: OBSERVATION OF USERS' BEHAVIORS

We conducted a Wizard-of-Oz study to explore the ideal experience of users in eyes-free voice-based editing and the difficulties they encountered. An experimenter disguised himself/herself as an automated editing software that allowed the user to easily and concisely enter and edit text through a voice interface and to complete the task of producing an easy-to-read paragraph. Participants were uninformed that a human experimenter performed the software.

3.1 Participants

Given that the language used in the study was Chinese, we recruited 10 native Chinese speakers (5 females, 5 males; age mean = 27.7, SD = 7.0) for the study. According to their self-reports, each was skilled in voice input and text editing but had no prior voice editing experience. The participants input and edited Chinese paragraphs and spoke Chinese commands.

3.2 Apparatus

Each participant wore a wired headset connected to a smartphone placed on a desk in front of them; the smartphone had a black screen. We developed an eyes-free voice-based text editor, which allowed participants to enter text verbally, listen to the recognized text and revise the incorrect text by voice commands.

The editor used buttons of the wired headset as a manual interface to navigate among text and distinguish between input and editing. The navigation unit was segment; each was a semantic unit separated by commas in the recognized text. Participants could click buttons to control the audio playback of recognized text or start playing from the previous or next segments. The editor allowed participants to stop the playback and state his/her editing intention when he/she found a mistake. In case of any editing command, the experimenter received the original text and editing command, revised the original text, and returned the final text result. The editor would read the returned result aloud for the participant and let him/her confirm or cancel the editing.

Table 1: The problems of original text in partial modification.

Problem	Description	Frequency	Example
Colloquial Insert	Unconsciously add words with no effect on the sentence meaning.	109	A User unconsciously adds *"So"* and *"um"* in the utterance *"So the weather is, um, pretty good"*.
Recognition Error	Part of a word or a phrase is missing, or some words are recognized as phonetically similar ones.	53	The utterance *"The cat sat."* was recognized as *"The bat sat."* or *"Cat sat."*
Improvable Expression	Some expressions are not accurate and complete, or users make some mistakes in their speech.	90	A User finds that the expression *"The weather is well"* is not good enough. It uses an improper adjective *"well"* and misses important information *"yesterday"*.

The editor was implemented as an Android application. We adopted Iflytek's Toolkit [2] for Automatic Speech Recognition (ASR) and Text-to-Speech (TTS).

3.3 Task and Procedure

Participants wrote two paragraphs of unspecified topics with our Wizard-of-Oz system, such as but not limited to telling a story, commenting on a hot event, or introducing a project; each paragraph with an average of 250 Chinese characters. During text input, participants played back the input text and edited it at free will. To modify a phrase in the input, they stopped the playback and informed the system how to modify it. There were no prior restrictions or explicit instructions on how to edit. Upon hearing the returned text result, the participant determined if the modification inclined to their expectation.

To cater for various editing operations in actual use, we asked participants to ensure that the finished paragraph was sufficient with no errors, easy to read, and unambiguous. Participants returned feedback on their experience in voice editing after completion of the tasks.

3.4 Data Collection

We recorded the audio of the speech interaction and interview. The editor recorded the original sentence, each modification and its editing command, the editing result, and the participant's feedback.

3.5 Results and Discussion

We collected 20 paragraphs, including 392 segments. The average length of segments was 10.94±4.80 Chinese characters. The average amount of participants' editing operations in a paragraph was 14.2±6.85.

3.5.1 Types of Editing. There were different types of editing since the problems existing in original sentences were different.

We labeled the categories of modifications and then counted the frequency of each category. In 37 editing operations (13%), participants re-spoke the whole segments to replace the original ones because complex problems occurred in the original segments. These segments could be disordered or logically confusing. Consequently, it was difficult for participants to determine which part was wrong. In other operations (87%, 247), participants edited a part of sentences because only one or several words needed to be changed. These partial modifications had three purposes:

(1) removing colloquial inserts (44.1%, 109);
(2) correcting recognition error (21.5%, 53);
(3) adjusting the wording (36.4%, 90) by replacing improper words, inserting some information or deleting unnecessary words.

2.0% of partial modifications had the second purpose and the third purpose at the same time. A detailed description of the problems mentioned in these purposes was shown in Table 1.

Excluding the first purpose, the other two purposes were achieved through three types of operations, i.e., insertion (21.0%, 29), replacement (57.2%, 79), and deletion (21.7%, 30).

It is essential to support all these modifications to design an effective voice interface for text input. According to these findings, we build a dataset to train and test the models that predicted the editing intentions, which Section 4.5.2 discusses.

3.5.2 Method of Editing. Participants used descriptive commands (13.7%), and just-speak-it commands (86.3%) alternately to edit by voice and prefer the later one.

The descriptive commands elaborated on the modification operation to be done, such as *"replace cat with bat"*. In a just-speak-it command, participants spoke a subpiece of the segment that contained the changed words. 43.3% of the participants only spoke about the changed words, and the rest added some context. 7 participants reported that editing by just speaking out the target words was intuitive. P5 said, *"The target words would come naturally to my mind and it was very natural and easy to say it."*

Nearly half of the just-speak-it commands do not contain context. This finding highlights the importance of locating replacement and insertion without relying on context.

3.5.3 Difficulty. During the interview, participants informed us about the difficulties of voice editing without visual feedback.

First, recalling the original text took up great energy. Consistent with previous research, participants pointed out the fatigue to use descriptive commands. Stating the error location required remembering the original incorrect words in replacement operations or recalling the context in insertion operations. Some errors were found and modified according to the whole segment, but after listening to the whole segment, especially when the segment was

[2] https://www.xfyun.cn/services/voicedictation

Table 2: The categories and examples of colloquial inserts. Red highlighted words are the target colloquial inserts to be removed.

Category	Subcategory	Frequency	Example (Translation)
Filled Pause	-	37	主人发现嗯，小猫在耍他。 The owner found out that, um, the kitten was playing with him.
Logical Confusion	Conjunction	19	然后这时候汤姆表示很喜爱杰瑞。 Then at this time Tom said he liked Jerry very much.
	Subject	10	有一个人他先进去了 One person he went in first.
User Habit	Modal Particles	24	比如顶橘子呀，踩滚轴呀等等。 Such as topping oranges ah, stepping on rollers ah and so on .
	Quantifiers	14	他把墙刷成了一个白色 He painted the wall one white.
	Pronouns	5	我们开始这个做一些工作。 We started this to do some work.

long, the details might be forgotten. Remembering the original text became more difficult if the original text included recognition errors that were difficult to understand, thus explaining why participants used descriptive commands less often.

Second, sometimes when using the re-speaking command, participants were confused about how much context they needed to add. Although our human imitated agent was more intelligent and more robust than the existing re-speaking algorithm, participants were uninformed that an experimenter did the editing. As a result, participants did not initially trust the system. Therefore, they did not know whether they needed to consciously add some context, worrying that the system could not understand their editing intentions based on their target words. Participants were concerned that they would quickly forget the initial context. These difficulties would bother the user if they used re-speaking commands in existing systems.

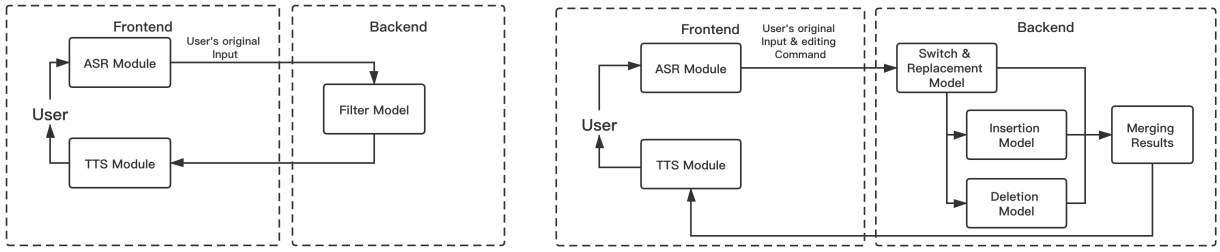
3.5.4 The Colloquial Inserts. Colloquial words were often unconsciously inserted into oral expressions. In this study, 9 participants habitually added these words, with 4 adding over 5 words into a single paragraph. A participant added 26 colloquial words to a paragraph. 38.4% of editing operations targeted the removal of these words, which was time-consuming. In the informal interview, participants reported that these operations were distracting and exhaustive. P3 said, *"Frequent deletions prevented me from concentrating on the embellishment of the text."* P4 said, *"There were so many of these mistakes that after changing them I no longer wanted to make any other changes"*. Due to the difficulty of simultaneous performance of these tasks and other modifications, 3 participants conducted 2 rounds of editing; removal of colloquial inserts in the first round and other adjustments in the second round.

Based in our analysis, we classified these inserts into three categories (shown in Table 2) with each category's frequency. Filled pause is a widely recognized category that users unconsciously added when they were thinking while speaking and did not want to finish their speech, such as *"well"*, *"um"*, etc. The functions of filled pauses include pausing and giving the speaker time to organize the language, making the tone more polite, expressing emotions, and others. Besides, we demonstrated two new categories of colloquial inserts. It was harder to identify them as pauses did not accompany these categories. The first category of words were inserted due to the logical confusion between or within segments. When inputting verbally, users had no way to carefully consider the logic between contexts like inputting with keyboard and visual feedback. So they might add some simple conjunctions ("就是" (so), "然后" (then), "不是" (but), etc.) unconsciously, or insert subjects when they are not needed. The second category of words were added by participants in fluent expressions because of their own linguistic habits, which were very common in Chinese but rarely appeared in English. Such as modal particles when emphasizing subjects or giving examples (express through tones in English), and pronouns or quantifiers ("一个" (one), "一些" (some), "这个" (this), "那个" (that)) in front of content words. These words usually occurred at the beginning and end of segments (67.3%), around nouns and verbs (27.3%), and at other places (5.5%).

3.5.5 Limitations of Voice Input. Participants also reported the limitations of eyes-free verbal text processing. Due to the limited memory space, one can not identify the logic between sentences as they can only focus on the local errors when listening to audio feedback. Therefore, it is more suitable for the input of short and medium texts without complex logical relationships or revising the entered text. Furthermore, although the ideal experience is to speak while conceiving, the speed of conceiving becomes slower when speaking and might not keep up with the speed of speaking. Thus, the recognized text will contain a lot of repetition, confusion, and consequent recognition errors, making the text difficult to edit. Hence, utilizing voice input requires users to have a preliminary idea before speaking.

3.5.6 List of Problems. In summary, we mainly discovered two problems for users in this study.

Problem 1 Colloquial inserts are common in Chinese oral expressions, which take up users' time and attention for editing.



(a) Automatic removal of colloquial inserts.

(b) Processing of just-speak-it commands.

Figure 2: System overview.

Problem 2 Users tend to just speak the target words to revise the text, including correcting recognition errors and improving expressions.

4 DEVELOPING A VOICE-BASED TEXT INPUT AGENT

Based on the findings in Study 1, we implemented SmartEdit, an eyes-free smart voice input assistant on the smartphone.

4.1 System Design

The goals of our implementation were to:

(1) Implement an agent that allows users to enter and edit text verbally without visual feedback;
(2) Design a system for the filtering of colloquial inserts to reduce operations and avoid distraction (for Problem 1);
(3) Enable the processing of just-speak-it commands so that users only need to focus on the target text of modification (for Problem 2).

When using just-speak-it commands, users do not need to concern about providing accurate and sufficient context to ensure that the system understands where they want to make changes. The recalling of the syntax of descriptive commands and wrongly recognized words are also unnecessary.

We implemented two baseline methods (descriptive command and re-speaking) to support the comparative evaluation and support modes exchange between our method and two baseline methods.

4.2 The Closed-loop Voice Input Process

We implemented a voice input agent in with an Android system. Users would take the buttons on the wired headset as a manual interface. Users could first long press the middle button and say, *"Someone, um, wants to buy a mouse."* The voice input agent would remove the colloquial word *"um,"* automatically. Secondly, users could short press the middle button to playback the sentence. When listening to the sentence, users might want to make changes. The editing operations included two steps: (1) long-pressing the minus button and said the target words *"The advertisement says"*; (2) hearing the result of automatic editing, *"The advertisement says someone wants to buy a mouse."*, and pressed the middle button to confirm the result. If the result was unsatisfactory, users could press the minus or plus buttons to listen to other results or press the middle button to cancel the editing. If he had input several segments, he could click the minus or plus buttons to select and playback the next or previous segments. The editing operation would be performed on the selected segment.

4.3 System Architecture

The user interface of our system is a voice-based input agent in a smartphone. The agent receives data information from voice input and controls information from buttons of a wired headset.

As shown in Figure 2, the frontend consists of an Automatic Speech Recognition module and a Text-to-Speech module. The backend receives original sentences and editing commands, deploys different algorithms to handle them, and sends back the results.

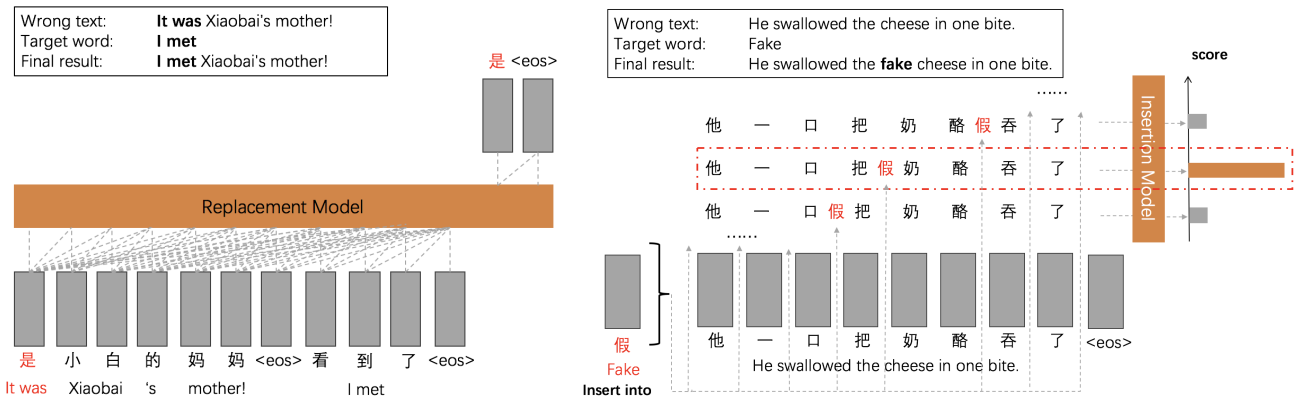
There are two states throughout the editing process: the results selection state and the common state. The former is the state between receiving the edit result and confirming it or canceling the edit, and the latter is the state during the rest of the time. The buttons serve different purposes in the two states (shown in Table 3).

Table 3: The functions of headset buttons.

Operation	Common State	Results Selection State
Long press middle button	Input	Cancel
Long press minus button	Edit	-
Press middle button	Playback & Pause	Confirm
Press minus button	Next Segment	Next Result
Press plus button	Previous Segment	Previous Result

4.4 Removing Colloquial Inserts

We implemented a neural model to remove colloquial inserts from the original sentences automatically. We considered the removing task as a sequence labeling problem where all colloquial inserts in the original sentences are tagged with 1 while others are tagged with 0. Referencing from the distribution characteristics described



(a) Replacement model's output is not empty meaning that the target words should replace the error span indicated by the generated results.

(b) The target word is inserted to each position of the original text, the best position is selected by the insertion model's language model score.

Figure 3: Example for determining the command type and position by the replacement model and the insertion model.

in Section 3.5, we created a dataset that randomly added colloquial words in the original sentences for the training and testing of the sequence labeling model. The source of the sentences was the chat data of the forum. Following Bidirectional Encoder Representations from Transformers (BERT) [6], We utilized a transformer-based model with a pooling layer added on the top of the last hidden layer of self-attention layers. Model parameters are initialized from BERT pre-trained weights. Besides, we built a real user dataset based on the data in Study 1. We collected common colloquial words and labeled the recognized text manually regarding the users' removal operations. Model performance on test dataset and real user dataset are shown in Table 4.

Table 4: Performance of colloquial words removal model on test dataset and real user dataset.

Dataset	Precision	Recall	F1
Test Dataset	87.83	87.93	87.59
Real User Dataset	98.37	92.36	95.28

4.5 Processing Just-Speak-It Commands

4.5.1 Overview. We treat the interactive error correction problem as a generative task in NLP.Formally speaking, for the given incorrect sentence W_s and just-speak-it command C_i provided by users as context, our system discriminates the modification type and locates the incorrect span in the provided sentence so that a corrected sentence C_s will be shown to users.

$$W_s + C_i \rightarrow C_s \tag{1}$$

There are three types of editing commands in interactive error correction: replacement, insertion, and deletion. The replacement operation replaces a consecutive span in the W_s with word or phrase provided by C_i, the insertion operation inserts word or phrase in C_i to a specific position in the W_s and deletion operation deletes a consecutive span in the W_s according to C_i.

Our system mainly consists of two models: the replacement model and the insertion model, as shown in Figure 3. The replacement model locates the error span in the given sentence and determines whether to use the insertion model instead of the replacement model. The insertion model selects the best position to insert the correct words into the given sentence. For each W_s and C_i, the replacement model is used in advance to generate a consecutive sequence in W_s, suggesting which part in W_s should be replaced, as seen in Figure 3(a). If the replacement model only generates the special token *end of sentence* (eos) without any content meaning that no span in W_s can be properly replaced by C_i, the insertion model will be used afterward to target the best insertion position, as seen in Figure 3(b). In practice, both the replacement model and the insertion model return more than one candidate, the probability of generating *end of sentence* token is used to balance the proportion of replacement candidates and insertion candidates shown to users.

Both replacement and insertion models inherit from Generative Pre-trained Transformer 2 (GPT-2) [20], a 12-layer transformer-based model with layer normalization and byte pair encoding [24] for the tokenizer.

Besides, we support the descriptive command for supplementation. We premise that users always add deletion keywords before the exact words they want to delete for deletion cases. Therefore, if an editing command starts with deletion keywords, our system returns the results by deleting the corresponding span in the error sentence without using the replacement or insertion models. When determining the deletion position, We use phonetic similarity instead of literal similarity to decrease the error introduced by automatic speech recognization (ASR). We add a special *"A 的 B"* (e.g. *"rice"* of *"eat rice"*, prevent *"rice"* from being recognized as *"raise"*) command to better describe B in the context of A so that we can deal with the case when B has homophones.

4.5.2 Data Preparation. Based on the error proportion shown in Section 3.5, we create the training dataset by simulating the real-time voice input scenario to train the replacement model and the insertion model. Specifically, first, we perform word segmentation on sentences collected from web forum posts because there is no explicit separator between words in Chinese. For the replacement case, we randomly replace the phrases in the sentence with phonetically or semantically similar phrases, consisting of consecutive words with random length. Phonetically similar phrases are generated by randomly changing the initials or vowels in phrases, and semantically similar phrases are generated by selecting from the top-ranked similar words for some words in the given phrase. Semantic similarity is measured by the inner product between embeddings of the phrase pairs. The embeddings of words are obtained by open-sourced Chinese word embeddings [25]. Missing or repeated words, synonyms, and words with similar pronunciation may appear in the same replacement phrase. In the insertion case, we randomly delete phrases in the original sentence and label the original sentence as a corrective sentence.

4.5.3 Training. Our training procedure consists of two steps. First, we pre-train our model on the Chinese-Wiki corpus with *language modeling loss* which is a commonly used left-to-right cross-entropy loss that learns the general language representations, then fine-tune it on our dataset to train the replacement model and the insertion model. The replacement model is trained to generate the incorrect span in the original incorrect sentence given W_s and C_i as context, while the insertion model is trained much similar to that in the pre-train step, which is only required to generate C_s ignoring W_s and C_i.

4.5.4 Model Evaluation. We evaluated the model performance through several tasks. First, as shown in Table 5 the *precision*, *recall*, and *F1* metric determined whether to switch to the insertion model by adjusting the threshold τ. Specifically, the replacement model will switch to the insertion model if the probability of $end\ of\ sentence$ token in the first decoding step is greater than τ. The replacement model performance among all replacement test cases is shown in Table 6. Recall at N (R@N) metric was used for different beam sizes. Table 7 shows the performance of the insertion model among all insertion test cases by R@N. Finally, our system was tested on a real user dataset collected in Study 2 to obtain the system's overall performance. It achieved a success rate of 94.7% when processing all the just-speak-it commands.

Table 5: Replace model performance for switching as hyper-parameter changes.

Threshold	Precision	Recall	F1
0.5	79.97	**80.18**	**80.08**
0.6	82.96	76.17	79.37
0.85	90.87	60.18	72.41
0.98	**98.20**	31.86	48.12

Table 6: Replace model performance as beam size changes.

Beam Size	R@1	R@3	R@5
1	86.32	-	-
3	**86.66**	**94.32**	-
5	86.62	94.12	**95.78**
10	86.61	93.78	95.54

Table 7: Insertion model performance on R@N.

Model	R@1	R@3	R@5
Insertion Model	85.25	96.36	**98.40**

4.6 Baseline Methods

We implemented two baselines for comparison: technique based on descriptive commands and re-speaking commands.

The technique based on descriptive commands provides different patterns for three types of instruction: *"replace A with B"*, *"insert A after B"*, and *"delete A"*. Each operation matches A in W_s by maximizing phonetic similarity.

The techniques based on the re-speaking commands inherit all the operations from the technique based on descriptive commands. Additionally, one can provide changed word or phrase along with several context words in the W_s for replacement and provide word or phrase along with both proceeding and the following text of the inserted position in the W_s for insertion. Specifically, if the beginning and end of a re-speaking command were aligned with a subpiece of the original text simultaneously, the subpiece would be used as an alternative alignment result.If only the beginning or end part was aligned with a subpiece of the original text, there would be two types of alternative results: replace the alignment text with the re-speaking text, or replace the alignment text along with the exact length text before or after the alignment position.

5 STUDY 2: EVALUATION

We conducted a within-subject user study to evaluate the usability of SmartEdit in practical tasks and compared it to agent based on descriptive command (DesCommand) and agent based on re-speaking command (Re-speaking). A participant would experience these three agents in three sessions.

5.1 Participants

We recruited 15 participants (9 females, 6 males; age mean = 24.5, SD = 1.6) for this study. All participants, according to their self-reports, were skilled in voice input, using voice assistants, and text editing, but none had previous experience in voice editing. The language in the study was Chinese. All participants were native Chinese speakers. None of these participants were involved in the previous study.

5.2 Apparatus

The participant performed the task wearing a wired headset connected to a smartphone with the screen turned off. The smartphone

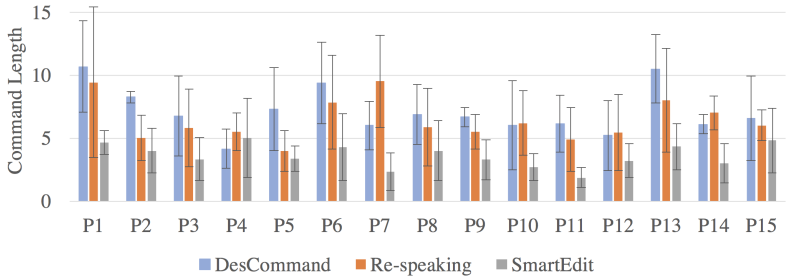


Figure 4: Average command length for different participants using different methods.

was placed on the desk in front of them. A voice input agent, as described before was installed in the smartphone. Based on SmartEdit, we had added DesCommand and Re-speaking as two modes in the agent. Descriptive commands were usable in Re-speaking and SmartEdit.

The interaction process of the three modes was similar. Colloquial inserts were automatically removed before a user edited the input text. Different algorithms were applied to calculate the candidate results after the original sentences and commands were sent to the server. The DesCommand supported three types of commands: insert, replace and delete. When using Re-speaking, if the re-speaking text aligned with the original text based on the pronunciation of characters, the system would replace the aligned text with the re-speaking text.

5.3 Task

In each session, the participant's task was to write a paragraph of about 250 Chinese characters to tell a prescribed story and then edit the text using a specified agent. The goal was to make the text of sufficient quality for sharing, which meant that it was error-free, easy to read, and unambiguous. Homophone errors could be ignored.

To reduce the difficulty of conceptualization for the participants, our task was designed as looking at pictures and talking. We found three cartoon videos that told simple stories with no or few lines. We made GIF pictures for the important plots, showing one plot per GIF picture (see Figure 5). In the experiment, we first let participants watch the entire video. Then, they looked at each GIF picture to describe the corresponding plot.

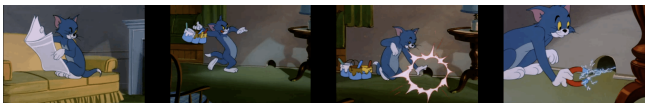


Figure 5: The pictures about important plots.

5.4 Procedure

At the beginning of each session, a participant received a 5-minutes tutorial that showed the best practice of the three agents. Then, the participant learned to use the three techniques by editing an example paragraph.

After finishing the tutorial, the participant needed to complete three tasks with three agents. Consequently, to a balanced Latin Square, the order to use these agents was defined in advance to achieve counterbalancing. After each task, the participant was asked to assess the task load. We used the raw Task Load Index (TLX) as the evaluation dimension, which was a variant of the Nasa TLX [4, 13] that dropped the pairwise competition and individual subscales less relevant to the task. Our study adopted 4 subscales: mental demand, physical demand, overall performance, and frustration level. The participant was asked to score for these subscales on a 5-point scale. Finally, we conducted an informal interview about the experience.

5.5 Data Collection

We recorded the audio of the speech interaction and interview. The voice input agent recorded all participants' operations, including the clicks of buttons, the recognized results of input texts and editing commands, the candidate results, and participants' choices, at the same time, recording each operation time. The server recorded the recognized text and the removed colloquial inserts.

5.6 Results and Discussion

Participants had performed 189 edits with DesCommand, 176 edits with Re-speaking, and 190 edits with SmartEdit. The commands of Re-speaking included 152 re-speaking commands and 24 descriptive commands. The commands of SmartEdit include 171 just-speak-it commands and 19 descriptive commands.

5.6.1 Length of Commands. We performed paired sample t-tests corrected with Bonferroni correction and found that the average length of the commands (amount of Chinese characters) in SmartEdit (4.24±1.40) was significantly shorter than that in DesCommand (7.07±1.88, $p<.01$) and Re-speaking (6.48±1.69, $p<.01$) (see Figure 4). Stating shorter commands are less likely to cause fatigue to the body, which lowers physical demands. P6 said, *"When the editing density is high, you can obviously feel that the simpler commands are easier to say."*

5.6.2 Failure Rate. According to paired sample t-tests corrected with Bonferroni correction, the average failure rate of SmartEdit (0.094 ± 0.023) is significantly lower than that of DesCommand

(0.234 ± 0.041, p<.05) and Re-speaking (0.205 ± 0.033, p<.05). SmartEdit has a significantly lower recognition error rate than DesCommand (p<.01).

We coded the causes of failure for further analysis. The results are shown in Figure 6. In DesCommand and Re-speaking, there were also errors caused by participants' fault. Participants might misremember the incorrect words or context in the original sentence. During the organization of commands, they might forget part of information such as target word, or make mistakes because the command was long and contained much information. These errors happened less often in SmartEdit because the users were less likely to recall the original text and used simpler the commands. Algorithm errors occurred in 4.7% of SmartEdit operations. The participant could add some context or use descriptive commands instead in the next attempt.

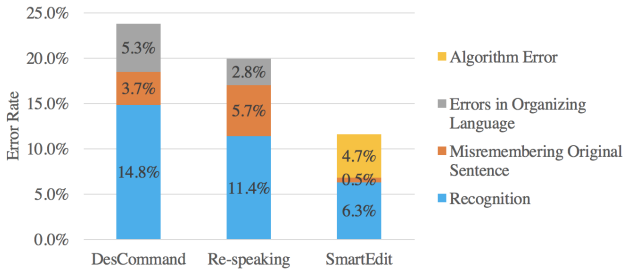


Figure 6: The rate of errors caused by various reasons.

Besides, The technique to determine the correct word among homophones by saying a phrase that contains the target word (e.g., *"Chi Fan De Fan"*, *"rice"* in *"eat rice"*) was effective enough to avoid introducing homophone errors when using short commands. 9 operations with just-speak-it commands in SmartEdit utilized this technique, and all of them succeeded.

5.6.3 Task Completion Time. The edit time is the total time taken when the participants pressed the button to start editing until the participant confirmed or rejected the result to finish editing. It might include the time of multiple attempts after failures.

We conducted paired sample t-tests corrected with Bonferroni correction on the average edit time and found that the edit time of SmartEdit (9.86±2.43 s) was shorter than that of DesCommand (12.67±3.38 s, p=0.15) and Re-speaking (12.98±5.50 s, p=0.17), but not significantly.

When the prediction is correct, editing time can be saved. However, the benefit is offset by the additional time spent selecting from the candidates or re-editing when the prediction is incorrect. SmartEdit (7.43±0.63 s) took more time to select results than DesCommand (5.94±0.46 s, p<.05) and Re-speaking (5.48±0.45 s, p<.05). Improving the accuracy of prediction is the key to improve the efficiency of SmartEdit further.

5.6.4 Effectiveness of Removing Colloquial Inserts. In the course of the experiment, the system removed a total of 212 colloquial inserts, of which 208 were correct. If the system did not automatically remove them, the participant would need to do 37.5% more editing operations. It was time-consuming and intermittently interrupted their thinking. The filter reduced the number of edits, thus improving the overall editing efficiency. Moreover, removing colloquial inserts also prevents the accuracy of the editing algorithm from being affected by them.

5.6.5 Subjective Experiences. Paired sample t-tests corrected with Bonferroni correction were performed on participants' subjective scores (shown in Figure 7). According to the result, the task load of SmartEdit was lower than that of DesCommand and Re-speaking.

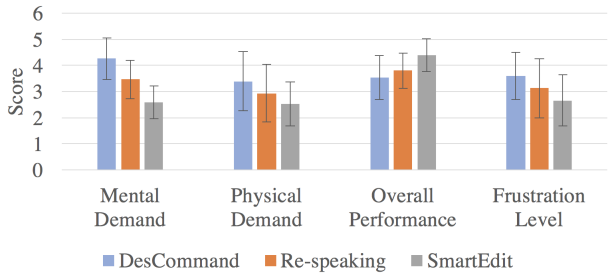


Figure 7: The average subjective scores for each scale of task load, including mental demand, physical demand, frustration level (lower is better) and overall performance (higher is better).

The mental demand of SmartEdit (2.60 ± 0.63) is significantly lower than that of DesCommand (4.27 ± 0.80 , p<.001) and Re-speaking (3.47 ± 0.74, p<.001).

The main mental burden of the DesCommand is to recall the original text and organize the complicated commands, while the SmartEdit method does not have these problems. The main mental burden of the Re-speaking is the concern about the context. 94.1% of the re-speaking commands include the context, and the remaining 5.9% of the re-speaking commands can be aligned with the original words by pronunciation. In contrast, when using SmartEdit, participants were not disturbed by considering context like using Re-speaking. As shown in Figure 8, they only spoke the changed words in 57.1% of the just-speak-it commands and unconsciously added some context in the rest 43.9% of tasks. In 15.6% of just-speak-it commands, the context was a word that could form a phrase together with the changed word. In 17.7% of just-speak-it commands, the context was several words around the changed text. In 9.5% of just-speak-it commands, participants re-spoke the whole segment. Participants only needed to think about what to modify and then instinctively add or not add context according to the situation. In the interview, P2 mentioned, *"When I heard error words, it would automatically come to my mind what words to change it into. Using the Re-speaking, I need to recall the original text because I am worried that the system cannot understand where I want to place it. But it is not needed when using SmartEdit."*

The physical demand of SmartEdit (2.53 ± 0.83) is significantly lower than that of DesCommand (3.40 ± 1.12, p<.01) and Re-speaking (2.93 ± 1.10, p<.05). Participants reported that the length of the command is a critical factor to the physical demand. The shorter commands in SmartEdit contributed to the lower physical demand scores.

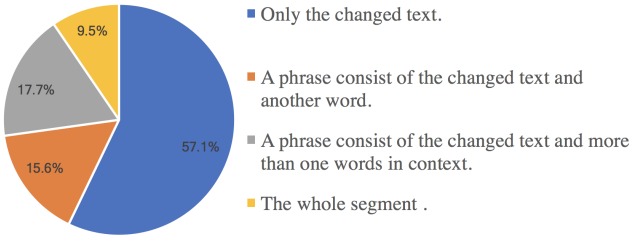


Figure 8: The percentage of just-speak-it commands with different length contexts.

With Re-speaking, participants sometimes avoided recalling accurate original text by including more context in the commands. 39% of the re-speaking commands said extra context, and 13% of re-speaking commands were the re-speaking of the whole segment. However, more prolix commands consumed more physical energy and even could lead to new recognition errors or inappropriate cohesion with the context.

SmartEdit (4.40 ± 0.63) had significantly better overall performance than DesCommand (3.53 ± 0.83, $p<.05$) and Re-speaking (3.80 ± 0.68, $p<.01$). The frustration level of SmartEdit (2.67 ± 0.98) is significantly lower than that of DesCommand (3.60 ± 0.91, $p<.01$) and Re-speaking (3.13 ± 1.13, $p<.05$).

Tasks with DesCommand and Re-speaking were more likely to fail. If participants failed, they had to go through multiple rounds to complete the modification. This process might cause frustration, and lower the participants' standards in the process, thereby reducing their satisfaction with performance. This explains the differences in the scores of overall performance and frustration level.

6 LIMITATIONS AND FUTURE WORK

In Study 2, we find that the algorithm's accuracy affects the efficiency and editing effort of SmartEdit. Algorithms that require less user input can reduce the cognitive load, but their accuracy may be lower. It is valuable to explore further the correlation between the accuracy of the algorithm and the user experience. In the future, we can introduce more language knowledge (e.g., grammar) to process just-speak-it commands more accurately. We can also leverage the language model to improve fault tolerance to process complex editing commands like re-speaking commands and descriptive commands.

Moreover, we can try to use short phrases instead of the whole segments as feedback to reduce the time spent in the selection of results. We can deeply research how much context is necessary for understanding.

Our smart voice assistant has mainly focused on the significant errors occurring during the voice input process, but has not covered some low frequency errors. Automatic Speech Recognition (ASR) may incorrectly recognize a word as its homophone. However, it is hard for users to find these errors through audio feedback. In addition, ASR probably makes mistakes in sentence segmentation, leading to the incorrect position of commas or other punctuation marks. Future research can explore the ways to correct these errors with the voice commands efficiently. We suggest:

(1) developing algorithms to find homophones and punctuation marks with a high possibility of error, and ask users whether they want to modify them;
(2) designing interactions to allow users to modify the punctuation, and check the spelling or characters of homophones.

Furthermore, to support various input tasks, voice input systems could also include definitions of new words in the future. The system will automatically remember the new words and gradually learn where it might appear.

7 IMPLICATION

Although our strategies bases on Chinese, they can be extended to English and other Germanic languages. Firstly, the pre-trained language models were proved effective in similar tasks for Germanic languages processing, such as automatic error corrections [15, 19] and named entity recognition [6]. Secondly, these languages do not introduce errors due to word segmentation, and the stem and the lemma of a word can assist in the alignment of the target words and the original text. However, since there are differences in the grammar and habits of different languages, verification should be performed before applying our strategies to other languages.

Our study implements a mobile voice input agent with a manual controller suitable for walking scenarios and accessibility. This approach is relevant in various other scenarios, such as in hands-free scenarios, like doing housework and driving, we can remove the manual control part and rely entirely on voice control. Voice commands support all operations that require headset buttons. Another extension is the voice input in virtual reality, augmented reality, large screens, and other visual scenes without a keyboard. Our approach will significantly shorten the length of the command and make the user feel laid-back. The modalities such as gaze and head movement can be combined to achieve higher accuracy.

8 CONCLUSION

Eyes-free voice editing is complex for users that work in mobile scenarios, home scenarios, and essential for visually impaired people. We implement two strategies to assist the editing. The first one is the automatic removal of colloquial inserts, which reduces tedious and repetitive editing tasks. The second one is editing by just speaking the target words, which prevents recalling and restating the original text. Our technique understands whether the user would like to insert or replace, at the same time where and how much he/she would like to change according to the target words, and it achieves a success rate of 94.7% in real tasks. We develop SmartEdit, a smart voice assistant based on the two strategies, allowing users to input and edit text by voice input. The evaluation of realistic text input tasks proves that SmartEdit can reduce 27.3% editing operations and is easier to use and less error-prone than previous techniques. The research contributes to our understanding of using state-of-the-art NLP approaches to minimize the cognitive load in eyes-free voice editing. We believe it is beneficial in eyes-free ubiquitous scenarios such as walking, driving, and accessibility.

ACKNOWLEDGMENTS

This work is supported by the National Key R&D Program of China No. 2019AAA0105200, and also by Beijing Key Lab of Networked Multimedia, the Institute for Guo Qiang, Tsinghua University, Institute for Artificial Intelligence, Tsinghua University (THUAI), and Beijing Academy of Artificial Intelligence (BAAI).

REFERENCES

[1] Shiri Azenkot and Nicole B. Lee. 2013. Exploring the use of speech input by blind people on mobile devices. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*. 11.

[2] S. K. Card, T. P. Moran, and A. Newell. 1987. *Computer text-editing: an information-processing analysis of a routine cognitive skill.* 219–240 pages.

[3] Junhwi Choi, Kyungduk Kim, Sungjin Lee, Seokhwan Kim, Donghyeon Lee, Injae Lee, and Gary Geunbae Lee. 2012. Seamless error correction interface for voice word processor. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 4973–4976.

[4] Lacey Colligan, Henry W.W. Potts, Chelsea T. Finn, and Robert A. Sinkin. 2015. Cognitive workload changes for nurses transitioning from a legacy system with paper documentation to a commercial electronic health record. *International Journal of Medical Informatics* 84, 7 (2015), 469–476.

[5] Jan Cu ́ı n, Martin Labsk, Tom ́a Macek, Jan Kleindienst, Hoi Young, Ann Thyme-Gobbel, Holger Quast, and Lars Knig. 2011. Dictating and editing short texts while driving: distraction and task completion. In *Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. 13–20.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[7] Qianqian Dong, Feng Wang, Zhen Yang, Wei Chen, Shuang Xu, and Bo Xu. 2019. Adapting Translation Models for Transcript Disfluency Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6351–6358.

[8] Ira A Gerson and Brett L Lindsley. 1989. Method for entering digit sequences by voice command. US Patent 4,870,686.

[9] Debjyoti Ghosh, Pin Sym Foong, Shengdong Zhao, Di Chen, and Morten Fjeld. 2018. EDITalk: Towards Designing Eyes-free Interactions for Mobile Word Processing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 403.

[10] Debjyoti Ghosh, Pin Sym Foong, Shengdong Zhao, Can Liu, Nuwan Janaka, and Vinitha Erusu. 2020. EYEditor: Towards On-the-Go Heads-Up Text Editing Using Voice and Manual Input. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 46.

[11] Debjyoti Ghosh, Can Liu, Shengdong Zhao, and Kotaro Hara. 2020. Commanding and Re-Dictation: Developing Eyes-Free Voice-Based Interaction for Editing Dictated Text. *ACM Transactions on Computer-Human Interaction* 27, 4 (2020), 1–31.

[12] Christine A. Halverson, Daniel B. Horn, Clare-Marie Karat, and John Karat. 1999. The Beauty of Errors: Patterns of Error Correction in Desktop Speech Systems.. In *INTERACT*. 133–140.

[13] Sandra G. Hart. 2006. Nasa-Task Load Index (NASA-TLX); 20 Years Later:. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50, 9 (2006), 904–908.

[14] Julian Hough and David Schlangen. 2015. Recurrent Neural Networks for Incremental Disfluency Detection. In *Interspeech 2015*. 849–853.

[15] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Vol. 1. 595–606.

[16] J. Lai and N. Yankelovich. 2006. Speech Interface Design. 764–770.

[17] Arthur E. McNair and Alex Waibel. 1994. Improving recognizer acceptance through robust, natural speech repair.. In *ICSLP*.

[18] George A. Miller. 1995. WordNet: a lexical database for English. *Communications of The ACM* 38, 11 (1995), 39–41.

[19] Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem N. Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – Grammatical Error Correction: Tag, Not Rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*. 163–170.

[20] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[21] Mohammad Sadegh Rasooli and Joel Tetreault. 2013. Joint Parsing and Disfluency Detection in Linear Time. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 124–129.

[22] Johan Schalkwyk, Doug Beeferman, Franoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Kamvar, and Brian Strope. 2010. "Your Word is my Command" : Google Search by Voice: A Case Study. (2010), 61–90.

[23] Korok Sengupta, Sabin Bhattarai, Sayan Sarcar, I. Scott MacKenzie, and Steffen Staab. 2020. Leveraging error correction in voice-based text entry by Talk-and-Gaze. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–11.

[24] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1715–1725. https://doi.org/10.18653/v1/P16-1162

[25] Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional Skip-Gram: Explicitly Distinguishing Left and Right Context for Word Embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 175–180. https://doi.org/10.18653/v1/N18-2028

[26] Matthias Sperber, Graham Neubig, Christian Fügen, Satoshi Nakamura, and Alex Waibel. 2013. Efficient Speech Transcription Through Respeaking. In *INTERSPEECH*. 1087–1091.

[27] Bernhard Suhm, Brad Myers, and Alex Waibel. 2001. Multimodal error correction for speech user interfaces. *ACM Transactions on Computer-Human Interaction* 8, 1 (2001), 60–98.

[28] Keith Vertanen and Per Ola Kristensson. 2009. Automatic selection of recognition errors by respeaking the intended text. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*. 130–135.

[29] Keith Vertanen and Per Ola Kristensson. 2010. Getting it right the second time: Recognition of spoken corrections. In *2010 IEEE Spoken Language Technology Workshop*. 289–294.

[30] Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided Pointer Networks for Chinese Spelling Check. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 5780–5785. https://doi.org/10.18653/v1/P19-1578

[31] Shaolei Wang, Wanxiang Che, and Ting Liu. 2016. A Neural Attention Model for Disfluency Detection.. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 278–287.

[32] James R. Williams. 1998. Guidelines for the Use of Multimedia in Instruction. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 42, 20 (1998), 1447–1451.

[33] Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency Detection Using a Bidirectional LSTM.. In *Interspeech 2016*. 2523–2527.

[34] Chengxiang Zhai and John Lafferty. 2001. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, Vol. 51. 334–342.

[35] Mingrui Ray Zhang, He Wen, and Jacob O. Wobbrock. 2019. Type, Then Correct: Intelligent Text Correction Techniques for Mobile Text Entry Using Neural Networks. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 843855. https://doi.org/10.1145/3332165.3347924

[36] Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling Error Correction with Soft-Masked BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 882–890. https://doi.org/10.18653/v1/2020.acl-main.82