

Diagnosing and Coping with Mode Errors in Korean-English Dual-language Keyboard

Sangyoon Lee

HCI Lab, School of Computing, KAIST
Daejeon, Republic of Korea
pinetree408@gmail.com

Jaeyeon Lee

HCI Lab, School of Computing, KAIST
Daejeon, Republic of Korea
jaeyeonlee@kaist.ac.kr

Geehyuk Lee

HCI Lab, School of Computing, KAIST
Daejeon, Republic of Korea
geehyuk@gmail.com

ABSTRACT

In countries where languages with non-Latin characters are prevalent, people use a keyboard with two language modes namely, the native language and English, and often experience mode errors. To diagnose the mode error problem, we conducted a field study and observed that 78% of the mode errors occurred immediately after application switching. We implemented four methods (Auto-switch, Preview, Smart-toggle, and Preview & Smart-toggle) based on three strategies to deal with the mode error problem and conducted field studies to verify their effectiveness. In the studies considering Korean-English dual input, Auto-switch was ineffective. On the contrary, Preview significantly reduced the mode errors from 75.1% to 41.3%, and Smart-toggle saved typing cost for recovering from mode errors. In Preview & Smart-toggle, Preview reduced mode errors and Smart-toggle handled 86.2% of the mode errors that slipped past Preview. These results suggest that Preview & Smart-toggle is a promising method for preventing mode errors for the Korean-English dual-input environment.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in HCI**; **Interaction techniques**.

KEYWORDS

Dual language keyboard; input language mode; mode error.

ACM Reference Format:

Sangyoon Lee, Jaeyeon Lee, and Geehyuk Lee. 2019. Diagnosing and Coping with Mode Errors in Korean-English Dual-language

Keyboard. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019)*, May 4–9, 2019, Glasgow, Scotland UK. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3290605.3300255>

1 INTRODUCTION

In countries such as Korea, Japan, China, and Thailand that use languages with non-Latin characters, people usually have two input language modes in computing devices: native language and English. Switching between the two input language modes occurs frequently while writing a document in both languages. Mode errors occur when a user's perceived input language mode is different from the computer's actual input language mode. Mode errors may not occur frequently when the user is entering text in a single application because the user is well aware of the current language mode as seen from the text that he/she is entering. On the contrary, mode errors may occur frequently when the user is switching between multiple applications owing to the following reasons. First, inputs in each application context may not be continuous; therefore, the user may forget the current language mode of the application. Second, there is a preferred language for each application (e.g., English for the input fields of a web URL and the native language for chat applications), which may bias the user's expectation regarding the current input language mode on an application switch. Input mode errors adversely affect user experience because they cost users additional efforts by erasing a wrong input and retyping in the right input language mode. Previous studies in Korea, Japan, China, and Thailand confirm that input language mode switching is error-prone and expensive [6, 8, 9, 12].

Figure 1 shows possible strategies to deal with the mode error problem. They can be divided into two categories depending on the role of the computer in them. In the 'Active' category, the computer plays an active role; it switches the input language mode automatically by predicting the user's intended language mode based on the past text inputs. In the 'Passive' category, the computer plays a passive role; it helps the user avoid mode errors by providing proper visual feedback; in other words, it reduces the cost of mode errors by providing an efficient recovery interface. Various implementations of some of these strategies already exist in the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

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

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-5970-2/19/05...\$15.00

<https://doi.org/10.1145/3290605.3300255>

| The role of the computer | Active | Passive | |
|--------------------------|-------------------------|----------------------------------|---|
| Strategy | Automatically switching | Providing proper Visual feedback | Providing an efficient Recovery interface |
| Method | Auto-switch | Preview | Smart-toggle |
| | | Preview & Smart-toggle | |

Figure 1: Strategies and methods to deal with the input language mode error problem.

Korean-English dual-language environment [6, 14, 15, 17]. However, they were designed mostly for use within a single application, and their usage and effectiveness in multi-application contexts have not been studied yet.

Based on these strategies, we designed the following four methods: Auto-switch, Preview, Smart-toggle, and Preview & Smart-toggle. Auto-switch switches the input language mode automatically based on the current text input and the language models of the two languages. Preview prevents the user from committing an error by providing clear visual feedback about the current language mode. Smart-toggle provides an interface to help the user recover from a mode error efficiently. The previous two methods can be used in combination. The user may commit a mode error despite Preview, and further may use Smart-toggle to correct the wrong input quickly. This combination is called Preview & Smart-toggle.

In this study, we first conducted a long-term field study (Experiment 1), to observe when and how often input language mode errors occur in the real context and thus understand the mode error problem quantitatively. We conducted four additional long-term field studies (Experiments 2 to 5) to verify the effectiveness of the aforementioned four methods.

The contributions of this study are as follows:

- We diagnosed the mode error problem in the Korean-English dual-input environment through a long-term field study.
- We examined the effectiveness of the aforementioned four methods through a series of long-term field studies and showed that Preview & Smart-toggle was most effective.
- We present a highly effective method (Preview & Smart-toggle) to deal with the language mode problem for the Korean-English dual-language environment.

In the remainder of this paper, we present a brief review of related work and describe the aforementioned four methods in detail. Next, we describe the design and results of the five experiments. Finally, we conclude by identifying the method that will be most effective for the Korean-English dual-language environment based on the results of the experiments.

2 BACKGROUND

In this section, we review the methods that have been used for users in the countries where languages with non-Latin characters are used. We further describe the Auto-switch method and review related prior work and commercial products. Finally, we review the existing methods for computers to provide users with the information about the current input language mode in dual-language input environments.

Dual-language Input Methods

In the countries where languages use non-Latin characters, there are two main approaches to enable dual-language text entry. The first approach is to allocate the characters of the native language to the keys of the QWERTY layout where English characters are already allocated. In other words, the keyboard has two overlapping layouts: the QWERTY(English) and the native language layout. One can switch between the two layouts using a toggle key. This approach is common when the native language uses phonogram characters, as Korean does.

The second approach is to use the QWERTY layout only. In the English mode, one uses the keyboard as usual. However, in the native language mode, to enter a native character, one uses its phonetically equivalent English input. Even in this case, the keyboard needs two language modes; therefore, a toggle key is used to switch between the two modes. This approach is common when the native language uses ideogram characters, as Chinese does. An example is the pinyin input method, which is one of the Chinese input methods. The pinyin input method uses the QWERTY layout; in the Chinese input mode, if the user inputs the pronunciation of a Chinese character in the English alphabet, a list of Chinese characters with a pronunciation matching the English input is displayed near the input area. When the user selects one of the proposed Chinese characters, the English input is converted to the Chinese character.

In the case of Japanese users, both the approaches are currently used. An example of the first approach is the kana input method. In this method, a Japanese key layout and the QWERTY layout share the keyboard, and users can switch between the two layouts using a toggle key. An example of the second approach is the romaji input method. In the Japanese input mode, when the user inputs the pronunciation of a Japanese character in the English alphabet, the input is automatically switched to the corresponding kana (Japanese syllabic scripts).

Automatic Mode Switching

Input language modes are necessary for a dual-language input method; whether it uses two keyboard layouts or phonetically equivalent English inputs. Switching between the

two language modes is usually done by a user action; however, it may also be done automatically by the computer by predicting the user's intended language mode.

In the Korean-English input environment, Ahn and Kang [6], Yoo et al. [17], and Lee [14, 15] proposed algorithms to predict the language of the current input keystroke based on the Korean and English corpora. In the Chinese-English input environment, Chen and Lee [9] showed that a new Chinese-English language model improved the prediction accuracy of automatic mode switching. In the Japanese-English input environment, Kasahara et al. [12] proposed a method for recognizing strings corresponding to Japanese words in inputted English characters and converting them into Japanese words. Ikegami et al. [11] proposed an algorithm for predicting input language mode, which was letter-based rather than word-based. Ehara and Tanaka [10] proposed an algorithm that enables an accurate prediction with a small corpus. In the Thai-English input environment, Chawannakul and Prasitjutrakul [8] and Potipiti et al. [21] proposed language models and algorithms for predicting the language mode of an inputted string.

Commercial products that provide automatic mode switching can be divided into two types. The first type provides the function within an application. MS Word is a popular example. MS Word provides an automatic mode switching between various languages and English. When the user enters a space character after entering a string, MS Word predicts the input language mode of the user based on the inputted string; if the predicted input mode is different from the current input mode, it converts the inputted string into a word in the predicted language [2]. The second type of products provides the auto-switching function through an input method editor (IME). ATOK [1], which is the most popular IME in Japan, and KeySwitcher [3], which provides auto-switching for various language combinations, are representative examples. ATOK displays an input in the other language mode near the text cursor if it determines that the input is entered in a wrong language mode. When the user selects the suggestion using a shortcut key, the current input is changed to the suggestion, and at the same time, the current language mode of the system is toggled. KeySwitcher provides auto-switching for all pairs of 24 languages including Russian. KeySwitcher predicts the current input language mode based on the user's input and indicates the predicted language mode with a flag icon. When the user accepts the language mode predicted by the KeySwitcher using a shortcut key, the inputted text is converted to text in the selected language, and the current language mode of the system is toggled.

The effectiveness and user acceptance of auto-switching seem to vary according to language. In Japan and China, both the first- and the second-type auto-switching products are popular [16]. In Korea, automatic mode switching is used

only within a single application such as a word processor; however, no commercial product providing automatic switching at the system level, such as through an IME, exists. The effectiveness and user acceptance of automatic switching in the Korean-English environment is questionable; it is one of the research questions that we tried to answer in this study.

Previewing Mode Errors with Visual Feedback

A mode error problem, in general, has long been an important topic in user interface research. A classic example is the vi editor in the Unix OS, which has two modes namely, the input and command mode. Poller et al. [20] described the occurrence of mode errors in the use of the vi editor in various environments. Norman describing a mode error as “doing the operation appropriate for one mode when in fact you are in another” [19]. A mode error occurs because the user cannot perceive the current mode of the system.

Shneiderman [23] advised providing informative feedback to deal with mode errors. Researchers explored the use of multi-modal feedback for preventing mode errors. Brewster et al. [7] provided a different audio feedback for each mode and found that such audio feedback reduces the user's mental load. Monk [18] found that audio feedback can significantly reduce mode errors. Sellen et al. [22] investigated the effect of visual and kinesthetic feedback on reducing mode errors.

Mode errors in dual-language input environments may also be reduced by proper feedback about the current language mode. Most of the existing GUI systems use an icon on the screen to represent the current input language mode. In the Mac OS, the current input mode is represented by a small icon in the status bar at the top of the screen. For example, in the English input mode, the icon shows the U.S. flag. The icon shows a national flag or the first character of a language to indicate the current language mode [4]. In the Windows OS, the current language mode is represented by a small icon in the status bar at the bottom of the screen. The icon shows the first character of a language or an abbreviation of a country name [5].

However, a small icon on the edge of the screen may not be effective feedback for reducing mode errors because it is often displayed away from the screen area where the user enters text. Therefore, a better method to provide visual feedback is desired to keep the user well aware of the current input language mode and reduce mode errors. In this study, we examine the use of visual feedback for reducing language switching mode errors, especially in multi-application contexts.

3 EXPERIMENT 1: DIAGNOSING THE LANGUAGE MODE ERROR PROBLEM

We conducted a long-term experiment to observe when and how often language mode errors occur in the real context.

Participants and Procedure

We recruited 30 participants (7 females, mean age = 21.7) through an online survey in our university. In the survey, we asked how candidates switched the language mode. We chose candidates who answered that they switched language modes frequently while using a computer. All the participants were university students; therefore, their computer usage patterns depended highly on their class timetables. Therefore, we set the experiment period to seven days to balance the effect of specific lectures or homework assignments. Additionally, we conducted the experiment during the semester while avoiding unusual periods such as midterm or final exams.

Before starting the experiment, each participant selected a computer with a physical keyboard that he/she used most frequently. 17 of the 30 participants selected laptop computers and the others selected desktop computers. We installed a hooking application that logs all key inputs on the selected computer. By analyzing the logged data, we obtained the key inputs that they made, the times, types, and the input modes of the key inputs, and the foreground applications during the key inputs. During the 7 days of the experiment period, the participants used their computers as usual. On completion of the experiment, we allowed the participants to remove sensitive data from the log files. We further collected the log files and analyzed the mode errors.

Metrics: Mode Errors and Mode Error Frequencies

Based on the definition of input mode error by Lee et al. [15], we consider a mode error to occur if the keyboard input pattern exhibits the following four phases: (typing, mode-switching, deleting, and retyping) or (typing, deleting, mode-switching, and retyping). In the typing phase, the user enters at least one keystroke excluding numeric and special characters. In the mode-switching phase, the user presses the language mode toggle key once. In the deleting phase, the user presses the backspace key a number of times. The number of the backspace key inputs may be greater than or equal to the number of the keystrokes entered in the typing phase. In the retyping phase, the user re-types the same keystroke sequence entered in the typing phase.

To quantify the mode error problem, in addition to the number of mode errors (*ME*), we counted the number of mode switches (*MS*), i.e., the number of language mode toggle key inputs. *MS* reflects the bilingual requirement of a computing environment. Generally, *ME* tends to increase as *MS* increases because the more frequently the user is exposed to language mode switching, the more frequently he/she may commit a mode error. In this respect, we defined mode error frequency (*MEF*) as the ratio between *ME* and *MS*: $MEF = ME/MS \times$

| | <i>MS</i> | | <i>ME</i> | | <i>MEF</i> [%] | |
|--------|-----------|------|-----------|------|----------------|-----|
| | Mean | SD | Mean | SD | Mean | SD |
| AS | 507.7 | 68.6 | 445.5 | 55.6 | 87.8 | 4.0 |
| Others | 250.0 | 69.2 | 123.5 | 59.8 | 49.4 | 3.7 |
| Total | 757.7 | 67.5 | 569.1 | 51.7 | 75.1 | 4.2 |

Table 1: Mean and standard deviations (SD) of *MS*, *ME*, and *MEF* in Experiment 1. ‘AS’ means that the values of *MS*, *ME*, and *MEF* consider only the mode switches and errors that occurred immediately following an application switch. ‘Others’ means that the values of *MS*, *ME*, and *MEF* consider other mode switches and errors.

100(%). *MEF* may reflect the relative severity of the mode error problem.

Results and Discussion

Table 1 lists the mean and standard deviations (SD) of *MS*, *ME*, and *MEF* during Experiment 1. On average, each participant made 757.5 mode switches and committed 569.1 mode errors during the experiment. This means that each participant experienced 81.3 mode errors every day on average, which shows the significance of the input mode error problem in the Korean-English dual-language input environment.

To examine the effect of application switching on mode errors, we counted the mode errors that occurred just after an application switch and those that occurred during the continuous use of an application; we counted the above separately and listed the results in the ‘AS’ and ‘Others’ rows, respectively. We looked at the first mode switches that occurred after an application switch, and observed that 98.9% of the first mode switches occurred within 3 minutes after an application switch. Based on this observation, we decided to count mode errors that occurred in 3 minutes after an application switch to obtain the numbers in the AS row. We could see that 78.3% of the mode errors occurred immediately after an application switch, and *MEF* could have reduced from 75.1% to 49.4% if the mode errors right after an application switch could have been prevented.

It was expected that the user would become more aware of the current input mode and would be less likely to commit a mode error as he/she spent time typing in a single application after an application switch. This seems to be supported by Figure 2, which shows *MEF* as a function of the elapsed time after an application switch (*TAAS*). The figure shows a clear inversely proportional relationship between *MEF* and *TAAS* ($R^2 = 0.97$).

It was expected that *MEF* may be dependent on application types because the time of use varies among different types of applications. We defined the time of use (*ToU*) in an application as the time between the first and last keyboard

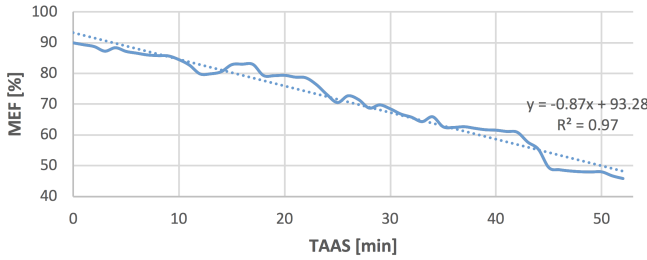


Figure 2: MEF vs. time elapsed after an application switch (TAAS) in Experiment 1.

| | MS | | ME | | MEF (%) | | ToU (min) | |
|-----|-------|-------|-------|------|---------|-----|-----------|-----|
| | Mean | SD | Mean | SD | Mean | SD | Mean | SD |
| TE | 243.4 | 65.5 | 178.3 | 48.6 | 73.2 | 4.3 | 21.9 | 1.5 |
| WB | 421.7 | 70.7 | 318.9 | 58.1 | 75.6 | 3.0 | 5.5 | 1.5 |
| SNS | 92.6 | 117.5 | 72.0 | 89.9 | 77.8 | 3.2 | 3.5 | 1.4 |

Table 2: Mean and standard deviations of MS, ME, and MEF of the three groups of applications in Experiment 1 (TE: text editors, WB: web browsers, and SNS: social network services).

inputs before an application switch. Table 2 shows *ToU*, *MS*, *ME*, and *MEF* values of three application groups, text editors (TE), web browsers (WB), and social network services (SNS). The TE group includes Hangul, MS Word, Vi, Eclipse, Visual Studio, Sublime Text, PyCharm, IntelliJ, MS Excel, MS PowerPoint, Sticker Memo, Processing, NotePad, and Memo; the WB group includes Safari, Chrome, and Internet Explorer; and the SNS group includes KakaoTalk, Telegram, Line, Slack, and iMessage. *ToU* was the shortest for SNS and was the longest for TE. *MEF* was highest in the SNS group and was lowest in the TE group. We could observe that there exists an inverse proportional relationship between *MEF* and *ToU*.

4 FOUR METHODS TO DEAL WITH LANGUAGE MODE ERRORS

In Experiment 1, we diagnosed when and how often the mode error problem occur in a real context. Particularly, we found that a mode error is likely to follow an application switch. Based on this understanding, we designed the following four methods: Auto-switch, Preview, Smart-toggle, and Preview & Smart-toggle. We describe them in detail in this section.

Auto-switch Method

User interaction with the computer in the Auto-switch method is illustrated in Figure 3. On an application switch (step a), the Auto-switch application, which implements the Auto-switch method at the system level, begins to store the user's keystrokes. In step b, the user enters keystrokes, 'thsu', and

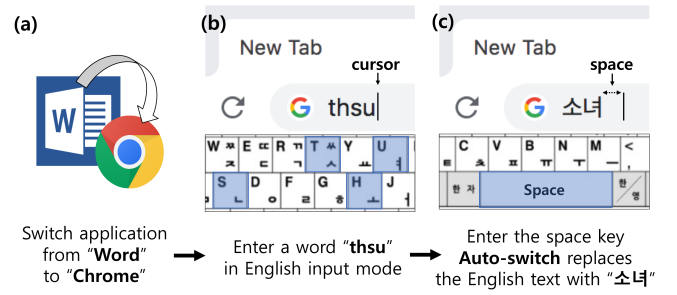


Figure 3: Interaction in Auto-switch: (a) Scenario after an application switch; (b) User enters a word; (c) User enters the space key.

the input field shows 'thsu', which is the English text string corresponding to the keystrokes because the current input mode is English. When the user presses the space key, the application predicts the language mode of the user based on the input entered so far. In this example, its prediction is different from the current input mode, and therefore it toggles the input mode and replaces the English text string with '소녀', which is a Korean text string corresponding to the same keystrokes.

In this example, the mode prediction looks correct, and the user will accept the changes by the application. If the mode prediction is not correct, the user can reject the changes using an undo function. In this case, the English word entered by the user will be added to a dictionary and will be used in the next prediction.

Mode Prediction Algorithm. The mode prediction algorithm used in Auto-switch is based on the algorithm proposed by Yoo et al. [17]. The algorithm predicts the language mode of the user when a space key is entered based on the non-number and non-special character keystrokes entered so far.

The keystrokes may represent a text string in English or Korean. First, the algorithm checks whether the English string is present in an English dictionary; it sets an English flag if it is present. Second, the algorithm checks whether the Korean string is present in a Korean dictionary; it sets a Korean flag if it is present. If both the flags are true or false at the same time, the algorithm does not do anything. If only one of the flags is true, then the algorithm outputs a prediction according to the true flag.

Our implementation of the algorithm used the Oxford English dictionary with 302,000 words and a standard Korean dictionary with 336,500 words. When we tested the algorithm with a test set used in other studies [6, 17], it showed a prediction accuracy of 98.5%. The accuracies of prediction

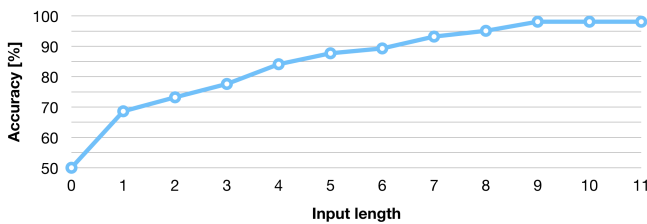


Figure 4: Expected prediction accuracy for different input lengths.

algorithms in other studies using the same test set are approximately 97%. We could confirm that our implementation was good enough for the current study.

Considered Interaction Designs. The Auto-switch application may intervene at different moments. We considered three different options. In the first option, the application predicts the user's input mode for every key input and intervenes when the predicted input mode is different from current input mode. In the second option, it predicts and intervenes after a predefined amount of input from the user. In the third option, it predicts and intervenes when the user finishes entering a word and presses a space key.

To choose the best option, we estimated prediction accuracies for different input lengths, and the results are shown in Figure 4. To estimate a language mode based on a complete word input, we used the mode prediction algorithm described in the previous subsection. However, to estimate a language mode based on an incomplete word input, we could not use the same algorithm because it requires a complete word input. Instead, we used the language mode prediction algorithm that was designed for the Smart-toggle method, which we will describe in Section 4. When input length is zero, the prediction accuracy is 50% because the prediction is between two languages. As input length increases, the prediction accuracy increases monotonically. For the first few letters, it is below 80%. The Auto-switch application may intervene frequently during a single word entry because its confidence about the user's input mode may oscillate between two languages. Frequent switching between two languages during a single word entry may result in a bad user experience. This seems to support that the first option would not be a good idea. To guarantee a stable prediction, the Auto-switch application needs to wait until at least 9 keystrokes are inputted. Based on this observation, we could have opted for the second option. However, considering that the average length of the words used by the prediction algorithm was 7.4 keystrokes, we chose the third option because the second option may not be very different from the third option in practice. In addition, the third option would enable an interface that is more predictable.

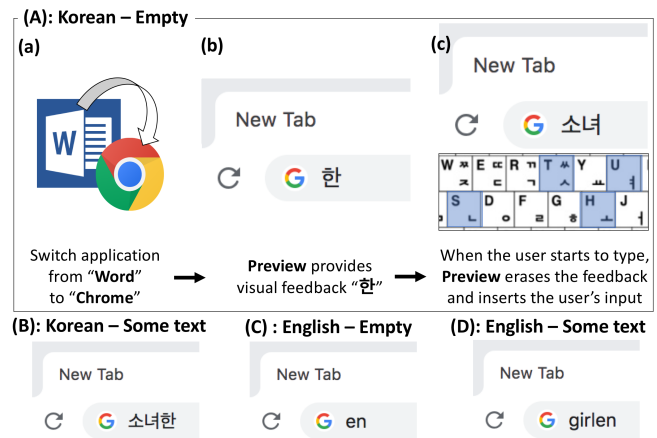


Figure 5: Interaction in Preview: (A) Case where the input field is empty in the Korean input mode; (B) Case where some text already exists in the input field in the Korean input mode; (C) Case where the input field is empty in the English input mode; (D) Case where some text already exists in the input field in the English input mode.

Our choice may be called ‘space-based’ auto-switching, and this option is in fact chosen by all auto-switching functions or applications available for the Korean-English dual-language environment. We speculate that their developers must have undergone the same considerations and simulations as we did.

Preview Method

The user interaction with the computer in the Preview method is illustrated in Figure 5. On an application switch, the Preview application, which implements the Preview method at the system level, provides visual feedback about the current language mode by temporarily inserting characters representing the current language mode into the current input field. It inserts ‘en’ to inform the user that the current input mode is English, or inserts the Korean character, ‘한’, to inform the user that the current input mode is Korean. When the user starts to type, the Preview application erases the visual feedback and inserts the user's input to the input field. Figure 5A – D shows the following four cases: the combinations of two input language modes (Korean and English) and two initial input field states (empty or non-empty).

Considered Visual Feedback Methods. We had considered three options to provide visual feedback about the current language mode to the user before we arrived at the final feedback design. The first option was to provide visual feedback by changing the mouse cursor. This option was not chosen because the mouse cursor is not always in the vicinity of an input field. The second option was to provide visual feedback

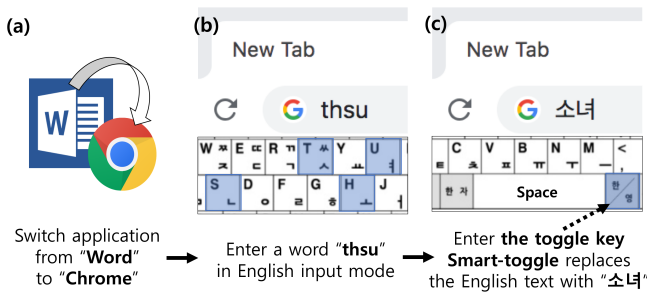


Figure 6: Interaction in Smart-toggle: (a) Scenario after an application switch; (b) User is typing; (c) User enters the toggle key.

about the current language input mode by changing the current text cursor. This option was not chosen, either, because it was not easy to change the visual aspect of the text cursor. The text cursor is not always controlled at the system level but is often controlled by individual applications.

The third option, which we finally chose, was to provide visual feedback about the current language input mode by inserting letters representative of the current language mode into the current input field. This method was implementable at the system level for all applications by generating virtual keyboard events. Additionally, this was effective for both Windows and Apple operating systems.

A problem with this option is that the characters inserted for visual feedback may be confused with the text entered by the user. We sought a method for making the characters inserted for visual feedback, visually distinct from the text entered by the user; however, we could not find a solution that works for all types of text input controls or widgets. 5 of 30 participants who participated in Experiment 3 actually mentioned this problem. However, we expected that the participants would be able to adapt to visual feedback and suffer less from the problem. To verify our expectation, we analyzed the logged data to count the participants' attempts to delete the visual feedback characters. The frequency of such attempts was approximately 10% initially; however, the frequency decreased gradually as the participants became accustomed to the interface. On the last day of the experiment, the frequency dropped to 1%. We will describe this analysis in more detail in Section 6.

Smart-toggle Method

User interaction with the computer in the Smart-toggle method is illustrated in Figure 6. On an application switch (step a), the Smart-toggle application, that implements the Smart-toggle method at the system level, starts to store the user's keystrokes. In step b, the user enters keystrokes, 'thsu', and the input field shows 'thsu', which is the English text string corresponding to the keystrokes because the current input

mode is English. In step c, the user notices a mode error and presses the *mode toggle key*. At this point, the application needs to predict the user's intention. The user may want to recover from a mode error and continue with the new mode, which is the case in this example, or may simply want to toggle the input mode from this moment on. The application computes the likelihood of the two language modes based on the user input: 'thsu'. The likelihood of the Korean mode is higher than that of the English mode in this case, and the application toggles the current input mode and replaces the English text string, 'thsu', with '소녀', which is the Korean text string corresponding to the keystrokes of 'thsu'. If the likelihood of the English mode were higher, the application would toggle the current input mode only without changing the English text string in the input field to the Korean text string.

The current Smart-toggle application starts to work on an application switch and stops when the user input exceeds 11 keystrokes. This design decision was based on an observation from Experiment 1; the first mode error after an application switching occurs within 11 keystrokes in 96.2% of the time. Working only until the first 11 keystrokes, the Smart-toggle application was expected to deal with most of the mode errors.

Mode Prediction Algorithm. A user may press the mode toggle key before a complete word is entered, and when the mode toggle key is pressed, the Smart-toggle application should make a prediction about the user's language mode. Therefore, we could not use the word-wise prediction algorithm used by the Auto-switch application. In fact, in the Korean-English dual-input environment, no prediction algorithm that works with a partial input is available. Therefore, we implemented a prediction algorithm that works with a partial input for ourselves.

This algorithm predicts a language mode based on the structural features of Korean and English characters and their dictionaries. In Korean, a character consists of 2 to 5 basic letters (consonants or vowels). For instance, a Korean character '가' consists of two basic letters, 'ㄱ' and 'ㅏ', and a Korean character '깍' consists of five basic letters 'ㄱ', 'ㅏ', 'ㅏ', 'ㅏ', and 'ㅏ'. To enter a character '가', the user presses two keys, 'ㄱ' and 'ㅏ', in the given order. It is important to note that not every possible sequence of basic letters is mapped to a valid Korean character. For example, two basic letters 'ㅏ' and 'ㄱ' in the given order are not mapped to any valid Korean character. In fact, it is very unlikely for a random sequence of basic letters to be mapped to a valid Korean character. An algorithm called Korean automata [13] can determine whether a given sequence of basic letters represents a valid Korean character or not.

| Entered | n | 4 | | | |
|---------|-----|------|------|------|------|
| | | 0 | 2 | 3 | 4 |
| Passed | m | | | | |
| | | | | | |
| P | e | 60.7 | 16.1 | 11.5 | 11.7 |
| | k | 0.0 | 0.0 | 43.7 | 56.3 |

Table 3: Probabilities for input text in English or Korean when four letters have been entered so far and m of those have passed Korean automata.

Table 3 lists the probabilities of the two language modes when n keystrokes (basic letters) have been entered so far and the first m of them have passed a Korean automata. The probability of the Korean mode (P_e) increases as m increases, whereas the probability of the English mode (P_e) decreases as m increases. These trends are consistent with our reasoning. The probability values in the table are determined by simulation. In the simulation, we counted the length of a sequence satisfying the automata when a part of a Korean or an English word was inputted. By repeating this for 336,500 Korean and 302,000 English words, we could build probability tables for different input lengths (n); Table 3 is one of them. In this table, the column for $m = 1$ is missing since there is no Korean letter consisting of one keystroke (basic letter). If the length of the inputted text so far (n) is 4, and if the number of letters that passed the Korean automata (m) is 3, the probabilities of the English mode (P_e) is 11.45 %, whereas the probability of the Korean mode (P_k) is 43.69 %. In this case, the probability of the Korean mode is higher; therefore the Smart-toggle application converts the input so far to Korean.

Preview & Smart-toggle Method

The Preview & Smart-toggle method is a sequential combination of the Preview method and the Smart-toggle method. The Preview & Smart-toggle application, which implements the Preview & Smart-toggle method, provides visual feedback about the current input mode, similar to the Preview application, and also provides the Smart-toggle feature. The Preview feature is expected to prevent the user from making a mode error. Nevertheless, the user may commit a mode error; in this case, the Smart-toggle feature enables the user to recover from the error efficiently.

5 EXPERIMENT 2: AUTO-SWITCH

We conducted a week-long field experiment to investigate the effectiveness of Auto-switch in the real context.

Participants and Procedure

We recruited 30 participants (6 females, mean age: 21.4) through the process described in Experiment 1. None of the participants had participated in Experiment 1.

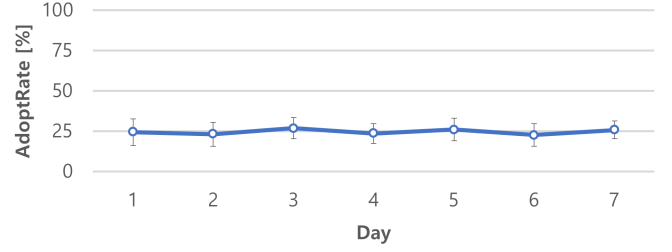


Figure 7: Trend of the *AdoptRate* of Auto-switch in Experiment 2. The error bars in the graph indicate standard deviation.

The overall procedure was the same as that of Experiment 1. 11 of 30 participants selected laptop computers, whereas the others selected desktop computers. Applications providing their own auto-switching feature were turned off. We conducted an interview with the participants on the last day about their experiences with Auto-switch.

Metrics: Adopting Rate and Accuracy

We were interested in how well the auto-switching feature was accepted and utilized by the participants. Therefore, we defined a metric called *AdoptRate*. We assumed that, if a user entered the space key after entering a word in a wrong mode, he/she had the intention to adopt the auto-switching feature. Therefore, we defined *AdoptRate* as $\frac{a}{a+b} \times 100[\%]$, where a is the number of the mode error cases where a user entered a space key after entering a word in a wrong mode to let the Auto-switch application correct the error. Further, b is the number of the mode error cases where a user deleted the inputted string before the Auto-switch application had a chance to correct the mode error.

We were also interested in how well the auto-switching feature could predict the input mode of the user correctly in a real-world context. We considered all the cases where the Auto-switch application predicted a language mode, and defined *Accuracy* as $\frac{a}{a+b} \times 100 (\%)$, where a is the number of cases where the Auto-switch application predicted a language mode correctly, and b is the number of cases where the prediction of the Auto-switch application was incorrect.

Results and Discussion

Figure 7 shows the average *AdoptRate* of Auto-switch during the experiment. The average *AdoptRate* over the whole period was 24.8%. To understand the reason for the low *AdoptRate*, we analyzed the keyboard input log of the cases where a mode error occurred. In 75.2% of the cases, participants noticed a mode error and started to correct the error before completing a word and pressing the space key; this means that Auto-switch did not have a chance to correct the error

in most cases. Participants could notice and correct a mode error after typing 2.54 keystrokes on average.

The average *Accuracy* of Auto-switch over the whole period was 94.8%. It is lower than the accuracy that we predicted based on the dictionaries (98.5%), which was described in the previous section. We speculate that the difference is due to proper nouns and neologism, which were not considered in the dictionary-based accuracy prediction. Although the actual accuracy of Auto-switch was slightly smaller than predicted, it is unlikely, that it was the main reason for the low *AdoptRate*.

In the post-experiment interview, approximately half of the participants (17/30) reported that they rarely used the Auto-switch function owing to the action of Auto-switch, which occurred after completing a word, which was often too late. The average length of the words used by the participants in this experiment was 7.56 keystrokes, whereas participants noticed and corrected a mode error after typing 2.54 keystrokes on average. This means that in most cases, the participants started to correct an error before the Auto-switch function had a chance to work. We speculate that such an early awareness of a mode error may be due to the clear visual differences between the Korean and English characters. This explains why an auto-switching method is not received well in the Korean-English environment, whereas it is received well in some of the other dual-language environments. We will discuss this interpretation in more detail in Section 9.

Based on the user comments and the analysis of the *AdoptRate*, we concluded that Auto-switch is not effective in dealing with the mode error problem in the Korean-English dual-language input environment.

6 EXPERIMENT 3: PREVIEW

We conducted another week-long field experiment to investigate the effectiveness of Preview in the real context.

Participants and Procedure

We recruited 30 participants (9 females, mean age: 21.0) through the same process as described in Experiment 1. None of the participants had participated in Experiments 1 or 2.

The overall procedure of Experiment 3 was the same as that of Experiment 1. 12 of 30 participants selected laptop computers, whereas others selected desktop computers. After the experiment, we computed the *MS*, *ME*, and *MEF*, using the same definitions used in Experiment 1.

Results and Discussion

Table 4 lists the mean and standard deviations (SD) of *MS*, *ME*, and *MEF* during Experiment 3. To compare the average *MEF* of Experiment 3 with that of Experiment 1, we used Welch’s t-test because the two participant groups ($N = 30$ for each

| | <i>MS</i> | | <i>ME</i> | | <i>MEF (%)</i> | |
|--------|-----------|------|-----------|------|----------------|-----|
| | Mean | SD | Mean | SD | Mean | SD |
| AS | 511.3 | 50.5 | 199.8 | 28.6 | 39.1 | 3.1 |
| Others | 268.9 | 49.3 | 122.1 | 30.8 | 45.4 | 2.6 |
| Total | 780.2 | 51.6 | 321.9 | 29.3 | 41.3 | 2.7 |

Table 4: Mean and standard deviations (SD) of *MS*, *ME*, and *MEF* in Experiment 3. ‘AS’ and ‘Others’ are the same as in Experiment 1.

| | <i>MS</i> | | <i>ME</i> | | <i>MEF (%)</i> | |
|-----|-----------|------|-----------|------|----------------|-----|
| | Mean | SD | Mean | SD | Mean | SD |
| TE | 241.6 | 90.8 | 92.2 | 34.0 | 38.2 | 3.3 |
| WB | 391.1 | 63.6 | 179.2 | 31.1 | 45.8 | 2.9 |
| SNS | 147.5 | 55.4 | 50.6 | 19.1 | 34.3 | 3.0 |

Table 5: Mean and standard deviations (SD) of *MS*, *ME*, and *MEF* for the three application groups in Experiment 3.

experiment) were independent and the normality condition was satisfied ($F = 1.1614$, $p > 0.05$). The test results showed that *MEF* difference between the two experiments was significant ($T(56) = 43.02$, $p < 0.001$). *MEF* decreased from 75.1% (in Experiment 1) to 41.3% (in Experiment 3). The decrease of *MEF* is even larger when only the AS case was considered (from 87.8% to 39.1%). This supports the conclusion that Preview was effective in reducing input mode errors in the Korean-English input environment.

Mode Error Reduction for Different Applications. Table 5 shows *MS*, *ME*, and *MEF* for three application groups in the experiment. To compare the average *MEF* between Experiments 1 and 3, we used the Welch’s t-test because the two participant groups ($N = 30$ for each experiment) were independent and the normality condition was satisfied as tested by the Shapiro-Wilk test ($F = 1.1614$, $p > 0.05$).

The test results showed that the averages of *MEF* from Experiments 1 and 3 were significantly different for the TE ($T(56) = 42.45$, $p < 0.001$), WB ($T(56) = 37.54$, $p < 0.001$), and SNS applications ($T(56) = 52.06$, $p < 0.001$). Compared with the results of Experiment 1, *MEF* decreased from 73.2% to 38.2% for the TE applications, from 75.6% to 45.8% for the WB applications, and from 77.8% to 34.3% for the SNS applications. The decrease of *MEF* was highest for the SNS applications. This agreed with our expectation because the SNS applications had smallest *ToU* as shown in Table 2. The SNS applications are therefore likely to suffer most from a mode error following an application switch among the three application groups.

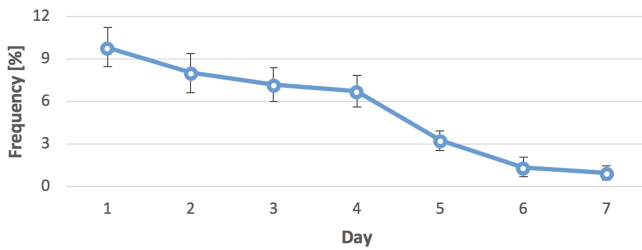


Figure 8: Frequency of the participants’ attempts to delete the visual feedback characters throughout Experiment 3. The error bars in the graph indicate standard deviation.

Adaptation to Visual Feedback. As aforementioned, we were concerned about the visual feedback method that we used. Participants were likely to confuse the letters injected for the visual feedback about the current input mode with inputted text. To check whether the participants were able to adapt to the visual feedback, we analyzed logged data and counted the participants’ attempts to delete the visual feedback characters. Figure 8 shows the frequency of such attempts during the experiment. On average, the frequency was 5.4%. The frequency of such attempts was approximately 10% initially; however, it decreased gradually. On the last day of the experiment, the frequency dropped to 1%. We speculate that participants could adapt to the visual feedback and were not disturbed appreciably by the less-than-ideal visual feedback, at least in the later phase of the experiment.

7 EXPERIMENT 4: SMART-TOGGLE

We conducted another week-long field experiment to investigate the effects of Smart-toggle in the real context.

Participants and Procedure

We recruited 12 participants (all male, mean age = 24.4) through the same process as described in Experiment 1. Nine of them had participated in Experiment 1. They were allowed to participate because Experiment 1 was not about a new method, but for observing existing computer usage practices. None of the 12 participants had participated in Experiments 2 or 3. After Experiments 1, 2, and 3, we realized that the number of participants could be reduced because the standard errors of *MEF* and *AdoptRate* were sufficiently small (Experiment 1: mean = 75.1, standard error = 0.8, Experiment 2: mean = 22.6, standard error = 0.4, and Experiment 3: mean = 41.3, standard error = 0.5). Therefore, we determined to conduct Experiments 4 and 5 with 12 participants for each.

The overall procedure of Experiment 4 was the same as that of Experiment 1. Three of 12 participants selected laptop computers and others selected desktop computers. We defined *AdoptRate* and *Accuracy* for the Smart-toggle method as we did for the Auto-switch method in Experiment 2. Their

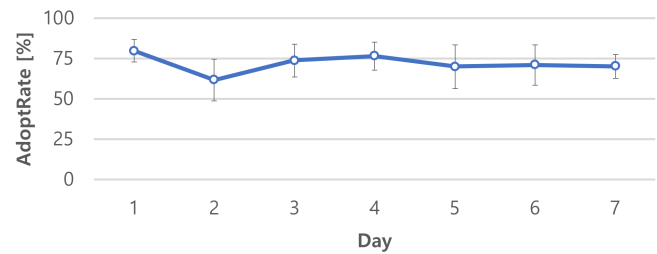


Figure 9: Trend of the *AdoptRate* of Smart-toggle in Experiment 4. The error bars in the graph indicate standard deviation.

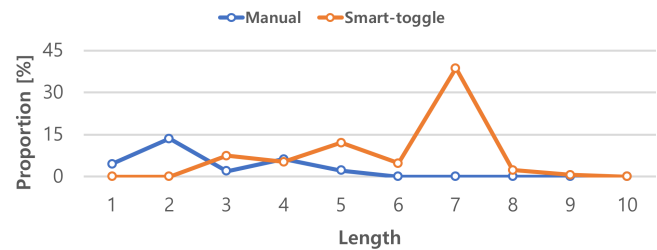


Figure 10: The proportions of the mode errors corrected manually or by Smart-toggle for different mode error lengths (in the number of keystrokes) in Experiment 4.

definitions are the same as before except for the definition of “adoption”. In the current case, we assumed that a user had the intention to adopt the Smart-toggle feature if he/she pressed the mode toggle key after entering some keystrokes in a wrong mode.

Results and Discussion

Figure 9 shows the average *AdoptRate* of Smart-toggle during the experiment. The average *AdoptRate* and *Accuracy* over the whole period were 71.4% and 93.1%, respectively. *AdoptRate* tended to converge after the first 3 days.

One of the participants mentioned that correcting manually (deleting and retyping text) was more convenient than using the Smart-toggle function for correcting a short error. Based on this comment, we analyzed the lengths of mode errors corrected by the Smart-toggle method and the lengths of mode errors corrected manually; the results are shown in Figure 10. For mode errors shorter than 4 keystrokes, participants used a manual method more often than the Smart-toggle method. On the other hand, for longer errors, participants used the Smart-toggle method more often than a manual method. Particularly, errors longer than 5 keystrokes, almost always used the Smart-toggle method.

We also analyzed how much typing could be saved by the participants with Smart-toggle method. Without Smart-toggle, they must have had to use the backspace key to delete letters entered in a wrong mode and retype in a correct

| Method | Exp | MS | | ME | | MEF (%) | |
|------------------------|-----|-------|------|-------|------|---------|-----|
| | | Mean | SD | Mean | SD | Mean | SD |
| Baseline | 1 | 757.7 | 67.5 | 569.1 | 51.7 | 75.1 | 4.2 |
| Auto-switch | 2 | 768.5 | 61.9 | 435.4 | 48.1 | 56.7 | 5.1 |
| Preview | 3 | 780.2 | 51.6 | 321.9 | 29.3 | 41.3 | 2.7 |
| Smart-toggle | 4 | 760.1 | 52.9 | 191.1 | 27.2 | 25.1 | 3.8 |
| Preview & Smart-toggle | 5 | 774.6 | 61.0 | 110.6 | 8.1 | 14.3 | 2.1 |

Table 6: Simulated Mean and standard deviations (SD) of MS, ME, and MEF values in all Experiments.

mode. During the experiment, each participant used Smart-toggle 551.6 times on average and could save 570.8 backspace keystrokes and 1655.3 retyping keystrokes on average.

8 EXPERIMENT 5: PREVIEW & SMART-TOGGLE

We conducted another week-long field experiment to investigate the effects of Preview & Smart-toggle in the real-world context.

Participants and Procedure

We recruited 12 participants (2 females, mean age: 24.6) through the same process as described in Experiment 1. None of the participants had participated in Experiments 1, 2, 3, or 4. The overall procedure was the same as that of Experiment 1. Four of 12 participants selected laptop computers and the others selected desktop computers. We conducted an interview on the last day about changes in their dual-language input experiences with Preview & Smart-toggle.

Results and Discussion

The average *MEF* over the whole period was 43.8%. The averages of *AdoptRate* and *Accuracy* over the whole period were 75.3% and 93.2%, respectively. We observed that *AdoptRate* converged after the first four days. On the last day of the experiment, 86.2 % of the mode errors which occurred despite the Preview method were corrected by Smart-toggle (*AdoptRate* = 86.2 %). 10.03% of the mode errors were corrected manually before Smart-toggle had a chance to correct them. Smart-toggle could not work for the remaining mode errors (3.74 %) because the length of the input was greater than ten.

These results show that Preview and Smart-toggle could play their respective roles in a complementary manner. User feedback in the post-experiment interview also confirmed that the two methods were complementary to each other.

9 DISCUSSION



Because the four methods that we considered in this paper were based on different strategies, we created and used metrics specific to each method. In this way we could show the benefits of each methods, but it was difficult to compare the four methods on the same ground. Therefore, we sought a

way to express the experiment results of the four methods in terms of the same metrics. In fact, it is possible to represent the results of all experiments in terms of *MS*, *ME*, and *MEF*. In Experiment 2, we excluded the mode errors that were handled by Auto-switch when we count *ME*. Likewise, in Experiment 4, we excluded the mode errors that were handled by Smart-toggle when we count *ME*. In Experiment 5, we excluded the mode errors that slipped past Preview but were handled by Smart-toggle when we count *ME*. Table 6 shows the resulting *MS*, *ME*, and *MEF* in all experiments. Based on these results, we concluded that Preview & Smart-toggle is the most effective (Simulated *MEF* = 14.3%) to deal with the mode error problem in Korean-English dual-language environment among the four methods.

This study was conducted in the Korean-English dual-input environment. The findings of this study may not translate directly to other dual-input environments. In particular, the relative effectivenesses of the four methods when they are applied to other dual-language environments may be different from what we observed in this study. The applicability of the four methods to other environments may primarily depends on whether the methods are dependent on language characteristics such as the statistical model of the language and its alphabet shape. If a method is dependent on language characteristics, our conclusions about it may not be valid in other dual-input environments. Answers to the effectiveness of the four methods in a different dual-input environment may be possible only by repeating the same experiments in that environment.

In a real-world text entry situation, users often generate typo. Because Auto-switch and Smart-toggle predict the current input language mode based on user input, the typo can affect the prediction result of the two methods. Suppose a user types “thsu” in the English mode when the intended input was “thsu”. This typo will not affect Smart-toggle because it does nothing until a user presses the mode toggle key. In this case, the user does not want to toggle the mode, and will use backspace keys to correct the typo. In other words, such a typo will not affect the performance of Smart-toggle. Suppose the same typo in the case of Auto-switch. A user will eventually enter a space, and then Auto-switch will switch the input mode to Korean, which in fact is not what the user wants. In other words, a typo may sometimes confuse Auto-switch and degrade its performance. It may be possible to estimate the size of the negative effect of typos on the performance of Auto-switch by comparing the language corpuses. If typical corrected error rates in the wild is given as well, it will be possible to predict how much the performance of Auto-switch will be affected by typos.

In the Preview method, we chose the method of injecting text characters into an input area to provide the visual feedback about the current language mode. This method was

effective for our study, but had a confusion problem, as we described earlier, although participants could overcome it in a few days. One of the possible ways to avoid the confusion problem would have been to inject graphical UNICODE characters, such as  and , instead of text characters. Such graphical UNICODE characters are clearly distinct from text input, and therefore the confusion problem could have been avoided. This method will be both effective and easy to implement in most GUI systems.

10 CONCLUSION

In this study, we observed that a majority (78.3%) of the mode errors occurred immediately after an application switch. We considered four methods to help users deal with mode errors that follow an application switch. Through a series of field studies in the Korean-English dual-input environment, we observed that Auto-switch was not accepted well, Preview could reduce the mode errors significantly (from 75.1% to 41.3%), and Smart-toggle was adopted well by users (71.4% of the time). We also observed that Preview & Smart-toggle was an effective complementary combination of Preview and Smart-toggle. We conclude that Preview & Smart-toggle is the best among the four methods for preventing mode errors for the Korean-English dual-input environment.

11 ACKNOWLEDGEMENT

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT (2017M3C4A7065963).

REFERENCES

- [1] 2017. ATOK. <http://www.justsystems.com/jp/products/atok/feature2.html?index=correction>
- [2] 2017. Automatically switch keyboard to match language of surrounding text. <https://support.office.com/en-us/article/Automatically-switch-keyboard-to-match-language-of-surrounding-text-c7316c3a-b6b3-433f-87f4-f777cf13decc>
- [3] 2017. KeySwitcher. <http://www.keyswitcher.com/Default.aspx>
- [4] 2017. Mac OS Input menu. https://support.apple.com/kb/PH22033?viewlocale=en_US&locale=en_US
- [5] 2017. Windows OS Keyboard Layout Language. <https://support.office.com/en-gb/article/Enable-or-change-a-keyboard-layout-language-1c2242c0-fe15-4bc3-99bc-535de6f4f258>
- [6] Young-Hoon Ahn and Seung-Shik Kang. 2001. Statistical approach to the automatic Korean-English string conversion. In *Proceedings of the 13th Annual Conference on Human and Cognitive Language Technology*. sigHCLT, Gyeongju, Korea, 205–208.
- [7] Stephen A Brewster, Peter C Wright, and Alistair DN Edwards. 1994. The design and evaluation of an auditory-enhanced scrollbar. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 173–179.
- [8] C. Chawannakul and S. Prasitjutrakul. 2011. Keyboard layout mismatch error detection and correction system utility. In *2011 8th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. 488–491. <https://doi.org/10.1109/ECTICON.2011.5947881>
- [9] Zheng Chen and Kai-Fu Lee. 2000. A new statistical approach to Chinese Pinyin input. In *Proceedings of the 38th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 241–247.
- [10] Yo Ehara and Kumiko Tanaka-Ishii. 2008. Multilingual Text Entry using Automatic Language Detection. *IJCNLP* (2008).
- [11] Y. Ikegami, Y. Sakurai, and S. Tsuruta. 2012. Modelless Japanese Input Method Using Multiple Character Sequence Features. In *2012 Eighth International Conference on Signal Image Technology and Internet Based Systems (SITIS)*. 613–618. <https://doi.org/10.1109/SITIS.2012.93>
- [12] Seiji Kasahara, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Error Correcting Romaji-kana Conversion for Japanese Language Education. *Proceedings of the Workshop on Advances in Text Input Methods* (2011), 38–42.
- [13] Injeong Kim and Heeyeung Hwang. 1989. A study on the Hangul automata. *The Transactions of the Korean Institute of electrical engineers* (1989), 491–494.
- [14] Keung-Hae Lee. 1995. Performance Evaluation of AIMS, an Automatic Korean / English Input Mode Switching System. In *Proceedings of the Korean Information Science Society Conference*.
- [15] Keung-Hae Lee. 1997. The Application Programming Interface of AIMS, A Predictive Input Mode Switching Technique. In *Proceedings of the Korean Information Science Society Conference*. kiise, Busan, Korea, 185–188.
- [16] Ken Lunde. 2008. Input Methods. In *CJKV Information Processing: Chinese, Japanese, Korean, and Vietnamese Computing*. "O'Reilly Media, Inc.", Chapter 5, 299–362.
- [17] Seung mok Yoo, Jeong hoon Lee, and Sam myo Kim. 1995. Automatic Korean / English Key Mode Converter. In *Proceedings of the Korean Information Science Society Conference*. 589–592.
- [18] Andrew Monk. 1986. Mode errors: a user-centred analysis and some preventative measures using keying-contingent sound. *International Journal of Man-Machine Studies* 24, 4 (April 1986), 313–327. [https://doi.org/10.1016/S0020-7373\(86\)80049-9](https://doi.org/10.1016/S0020-7373(86)80049-9)
- [19] Donald A. Norman. 1983. Design Rules Based on Analyses of Human Error. *Commun. ACM* 26, 4 (April 1983), 254–258. <https://doi.org/10.1145/2163.358092>
- [20] Merle F Poller and Susan K Garter. 1984. The effects of modes on text editing by experienced editor users. *Human Factors* 26, 4 (1984), 449–462.
- [21] Tanapong Potipiti, Virach Sornlertlamvanich, and Kanokwut Thanadkran. 2001. Towards an Intelligent Multilingual Keyboard System. In *Proceedings of the First International Conference on Human Language Technology Research (HLT '01)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1–4. <https://doi.org/10.3115/1072133.1072222>
- [22] Abigail J. Sellen, Gordon P. Kurtenbach, and William A. S. Buxton. 1992. The Prevention of Mode Errors Through Sensory Feedback. *Hum.-Comput. Interact.* 7, 2 (June 1992), 141–164. https://doi.org/10.1207/s15327051hci0702_1
- [23] Ben Shneiderman, Catherine Plaisant, Maxine S Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. 2016. *Designing the user interface: strategies for effective human-computer interaction*. Pearson.