

卒業論文 2024 年度(令和6年度)

入力コンテキストを切り替えることによる 日本語入力高速化手法の提案

慶應義塾大学 環境情報学部

石川湧馬

増井俊之研究会

2025年1月

入力コンテキストを切り替えることによる 日本語入力高速化手法の提案

論文要旨

現状の日本語入力では、同音異義語や煩雑な変換フローに起因するユーザー負担の増大という課題感がある。

そこで本研究は、入力コンテキストを切り替えることで変換候補を最適化し、入力効率を向上させるシステムを提案する。これにより、変換ミスの低減と入力手間の削減、定型文の高速入力を実現する。

キーワード

日本語入力, IME, ユーザーインターフェース

目次

第1章 はじめに	1
1.1 研究背景	1
1.2 研究目的	1
1.3 本稿の構成	1
第2章 既存システムとその課題点	2
2.1 従来の日本語入力フロー	2
2.2 入力負荷の問題	2
2.2.1 同音異義語の問題	2
2.2.2 定型文入力の問題	3
第3章 提案手法の概要	5
3.1 コンテキスト切り替えの概念	5
3.2 コンテキストの登録	5
3.3 入力方法	5
第4章 システムの提案	6
4.1 開発環境と技術要件	6
4.2 インターフェース概要	6
4.3 プロトタイプの機能構成	7
4.3.1 コンテキストの切り替え	8
4.3.2 キーとなる文字列の入力と候補の選択	9
4.4 コンテキストの実装	10
第5章 評価と考察	11
5.1 入力速度の評価	11
5.2 コンテキスト切り替えの議論	11
5.3 コンテキストのロジックの議論	11

5.3 コンテキスト作成コストの議論	12
5.4 別のアプローチとの比較	12
5.4.1 MacOSのユーザー辞書	12
5.4.2 テキストエクスパンダー	13
5.4.3 個別分野に特化したIME・辞書	15
第6章 結論	16
6.1 成果と知見	16
6.2 今後の展望	16
6.3 結論	16
謝辞	17
参考文献	18

図目次

2.1 MacOSの日本語入力フロー 「ありがとう」	4
2.2 同音異義語の変換例 「後攻」	5
2.3 定型文入力の例 「お世話になっております」 (都度変換)	6
2.4 定型文入力の例 「お世話になっております」 (一括変換)	6
4.1 システムインターフェース	8
4.2 データフロー	10
4.3 コンテキスト切り替えの様子	11
4.4 入力中の文字列と変換候補	11
4.5 コンテキストの実装の例	12
5.1 ユーザー辞書の活用 辞書の画面	15
5.2 ユーザー辞書の活用 候補の様子	15
5.3 espansoの例	16
5.4 espanso候補の選択	16

第1章 はじめに

現状の日本語入力では、同音異義語やかな漢字変換フローに起因するユーザー負荷の増大という課題感がある。そこで本研究は、入力コンテキストを動的に切り替えてからユーザーに入力を開始してもらうことで変換候補を最適化し、入力効率を向上させるシステムを提案する。

1.1 研究背景

従来の日本語入力システムは、多数の候補から目的の言葉を選択する必要があるため、ユーザーに負担をかける場合がある。特に使う語彙が限られた業務上のやりとりなどでは、明らかに入力しないであろう語が選択肢に含まれてきてしまったり、単語が長く系統的に絞り込みは簡単にできるにも関わらずユーザーは全て正しくタイピングを行わないと変換できなかったりする。このようなシステムはユーザーの入力負担が大きく、思考を妨げる要因ともなっている。この問題に対処するため、新たなアプローチとして、入力時のコンテキストをに事前に切り替えることで、変換候補を最適化する手法を提案する。

1.2 研究目的

入力コンテキストの切り替えによる日本語入力的高速化と誤変換の低減を主眼に置き、プロトタイプを開発し評価を行う。評価範囲としては、主に文章入力シーンを想定し、既存システムとの比較して評価を行う。

1.3 本稿の構成

第2章では既存システムの課題を整理し、第3章で提案手法を概説する。第4章で実装したプロトタイプと機能、第5章で評価と考察を示す。最後に第6章で本研究の総括を行う。

第2章 既存システムとその課題点

本章では、従来の日本語入力システムの問題点を整理し、提案手法の背景となる課題を明らかにする。

2.1 従来の日本語入力フロー

MacOSに標準搭載されている日本語入力システムを例に取り、従来の日本語入力フローを説明する。一般的な日本語IMEの操作手順 「ありがとう」という単語を入力する場合を例にとる。以下は左からa,r,iと入力した時のIMEの変換候補の図である。

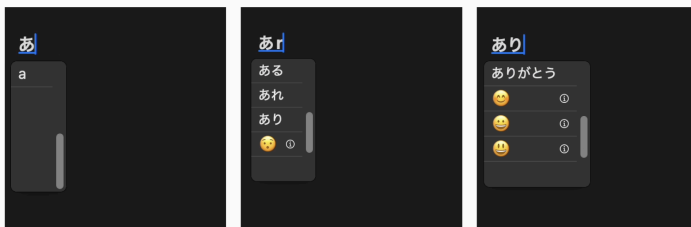


図2.1 MacOSの日本語入力フロー 「ありがとう」

「あり」が入力完了した時点で候補に「ありがとう」が表示された。スペースキーまたは下矢印キーを押すことで候補を確定し、変換が完了する。

2.2 入力負荷の問題

既存の日本語入力システムにおける課題点として、同音異義語と定型文入力の問題が挙げる

2.2.1 同音異義語の問題

「ありがとう」のような簡単な単語では問題はないが、複数の候補が表示される場合や、同音異義語が多い場合、ユーザーは候補を選択するために多くの操作を行わなければならない。

「後攻」という単語を入力する場合を考える。MacOSでは「こうこう」と入力すると、以下のような候補が表示される。



図2.2 同音異義語の変換例「後校」

この場合、ユーザーは候補を選択するために、スペースキーを押すか、下矢印キーを押して候補を選択する必要がある。今回であれば「こうこう」と入力してからスペースキーを7回押すことで「後校」を選択することができたが、このような操作はユーザーにとって負担が大きく、思考のスピードと日本語入力との間に大きく乖離が生じる。

これにより、入力効率が低下し、ユーザー負担が増大する。また、変換ミスが発生するリスクも高まる。

2.2.2 定型文入力の問題

また、定型文や挨拶文など、特定の文脈で頻繁に使用されるフレーズを入力する際にも、従来のIMEではその文章の読み方を入力する必要があるので、長い文章を入力したい場合には必然的に必要な工数が増加する。

「お世話になっております」というフレーズを入力する場合を考える。この場合、ユーザーは「おせわになっております」と入力する必要があるが、このような長いフレーズを入力する際には、ユーザーは入力に時間を要する。

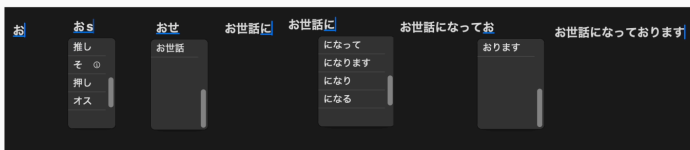


図2.3 定型文入力の例「お世話になっております」(都度変換)

「おせ」の時点で変換に「お世話」が表示されたので選択、その後に「に」を入力すると文脈を読んで「お世話」に接続する「に」から始まる候補として「になって」が表示されるので選択、その後に「お」を入力すると「お世話になって」に接続する「お」から始まる候補として「おります」が表示されるので選択し「お世話になっております」の入力が完了した。

文脈を読んで候補を表示するIMEの機能は非常に優れているが、このような長いフレーズを入力する際には、ユーザーは入力に時間を要する。

画像のような形態素ごとに変換を確定していく方法とは別で、「おせわになっております」と入力してからスペースキーを押すことで一気に変換する方法もあるが、この場合もユーザーは全ての読みを入力する必要があるため、入力効率が低下する。また、タイプミスリスクも高まるので入力効率の期待値は低い。

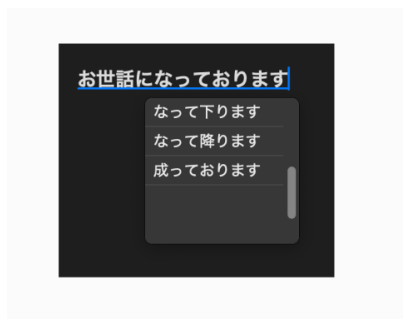


図2.4 定型文入力の例「お世話になっております」(一括変換)

第3章 提案手法の概要

第2章で整理した課題に対処するため、入力コンテキストを切り替えることで変換候補を最適化する手法を提案する。

3.1 コンテキスト切り替えの概念

まず、本稿におけるコンテキストとは何かを説明する。コンテキストとは、入力時における文脈や状況を指す。例えば、メールの文面を入力する際と、プログラムのコードを入力する際には、使用する語彙や文法が異なる。このような文脈を切り替えることで、IMEが適切な変換候補を表示することが可能となる。コンテキスト粒度は自由に設定可能なものであり、ユーザーが任意のコンテキストを登録し、切り替えることができる。「メール文面」と「プログラムコード」のような荒めのコンテキストから、「挨拶文」と「業務報告文」のような細かいコンテキスト、さらには「都道府県名」や「プロジェクト名」のような極めて細かいコンテキストまで設定可能な概念である。

3.2 コンテキストの登録

ユーザーは自分が使いたいコンテキストを事前に登録しておくことができる。登録しておいたコンテキストは、IMEの入力時に切り替えるための操作をすることで入力候補がそのコンテキストに合わせて表示されるようになる。

3.3 入力方法

変換ロジックが大きく異なるが、UIは従来のもとは大きく変わらない。ユーザーは現在のコンテキストの候補に入力することでそのコンテキストにおける候補一覧ロジックから出力された候補が表示される。ユーザーはその候補を選択することで変換を行うことができる。

第4章 システムの提案

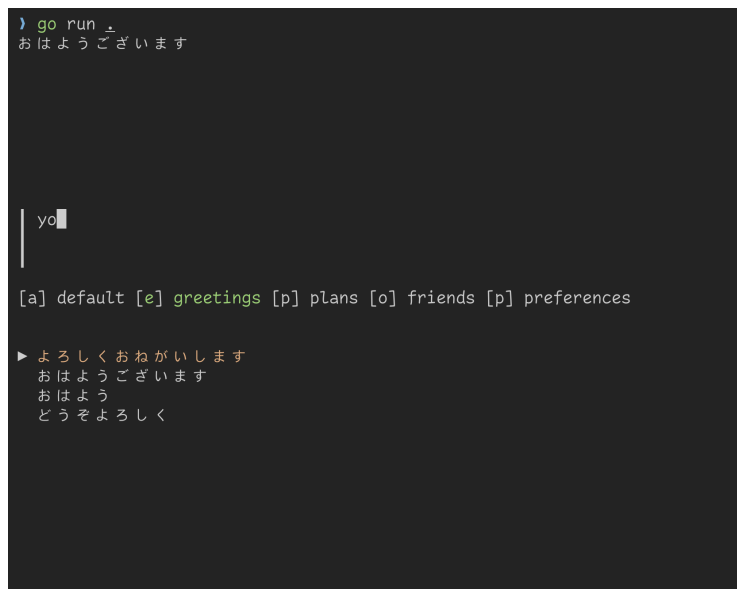
提案手法を検証するために開発したプロトタイプの開発環境や主要機能について概説する。

4.1 開発環境と技術要件

Go言語のTUIフレームワークであるbubbletea[1] とそのコンポーネントであるbubbles[2] を用いてターミナルで動作する今回提案するシステムのプロトタイプを開発した。Go言語はクロスプラットフォームで動作するため、Linux, MacOS, Windowsなどの環境で動作することが可能である。今回はMacOS Sequoia 15.1.1 で開発と動作確認を行った。

4.2 インターフェース概要

プロトタイプのインターフェースは以下のようになっている。



```
> go run .  
おはようございます  
  
yo  
  
[a] default [e] greetings [p] plans [o] friends [p] preferences  
  
▶ よろしくおねがいします  
  おはようございます  
  おはよう  
  どうぞよろしく
```

図4.1 システムインターフェース

1. 入力を確認した文字列を表示するエリア。
画像では「おはようございます」と入力した場合の画面を示している。
2. 現在入力中の文字列を表示するエリア。
画像では「yo」と入力中の場面を示している。
3. コンテキストの一覧と現在のコンテキストを表示するエリア。
画像では「[e] greetings」というコンテキストを選択している。
「[e]」というのはコンテキストのショートカットキーであり、
「greetings」というのは挨拶文コンテキストであることを示している。
4. 現在のコンテキストにおける変換候補を表示するエリア。
画像では「よろしくお願いします」「おはようございます」「おはよう」「どうぞよろしく」の4つの候補が表示されている。

4.3 プロトタイプの機能構成

アプリケーションは起動後に以下のことが可能となる

1. コンテキストを切り替える
2. キーとなる文字列を入力する
3. 入力表示されたキーを元に候補から選択する

以下はデータフローの概要である。

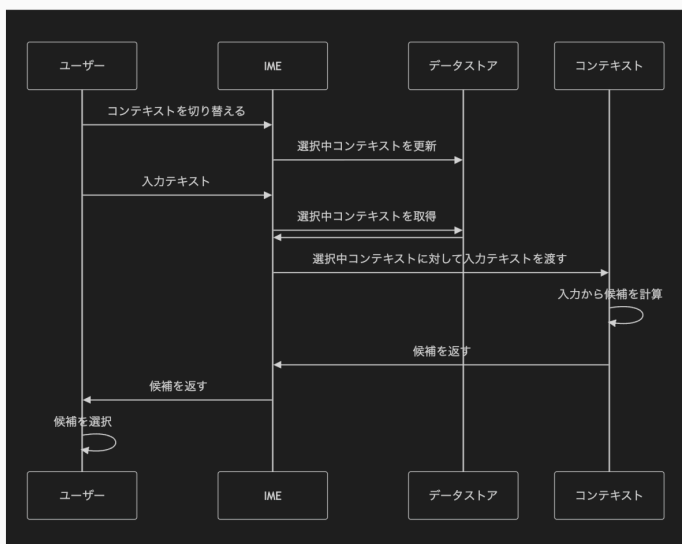


図4.2 データフロー

4.3.1 コンテキストの切り替え

コンテキストの切り替えは、ctrl+[登録されたキー]または左右の矢印キーで行うことができる。以下はコンテキストの切り替えの様子である。画像左側の「[p] greetings」を選択している状態で「ctrl+p」を押すことで、画像右側の状態に切り替わる。



図4.3 コンテキスト切り替えの様子

4.3.2 キーとなる文字列の入力と候補の選択

キーとなる文字列の入力は、従来のIMEの入力と同様に行うことができる。入力中の文字列は画面中央のエリアに表示される。入力中の文字列を元に変換候補を表示され、ユーザーはその中から選択することで変換を行うことができる。

以下は「お世話になっております」と入力する際の様子である。



図4.4 入力中の文字列と変換候補

「os」と入力した時点で「お世話になっております」が変換候補として表示され、下矢印キーまたはctrl+jキーを入力し、エンターキーを押すことで

変換の確定を行うことができる。

4.4 コンテキストの実装

入力された文字列を元に、コンテキストに応じた変換候補を表示するために、JavaScriptを用いてコンテキストのロジックを実装した。以下は greetings コンテキストのロジックの例である。入力したい文字列とローマ字の読み両方にマッチする候補を表示する単純なロジックを実装している。

```
1 const dict = [
2   { name: 'ありがとうございます', yomi: 'arigatougozaimasu' },
3   { name: 'よろしくおねがいします', yomi: 'yorosikuonegaishimasu' },
4   { name: 'ごめんなさい', yomi: 'gomennasai' },
5   { name: 'すみません', yomi: 'sumimasen' },
6   { name: 'おはようございます', yomi: 'ohayougozaimasu' },
7   { name: 'お世話になっております', yomi: 'osewani natteorimasu' },
8   { name: 'おはよう', yomi: 'ohayou' },
9   { name: 'こんにちは', yomi: 'konnichiwa' },
10  { name: 'こんばんは', yomi: 'konbanwa' },
11  { name: 'おやすみなさい', yomi: 'oyasuminasai' },
12  { name: 'はじめまして', yomi: 'hajimemashite' },
13  { name: 'どうぞよろしく', yomi: 'douzoyoroshiku' },
14  { name: 'おめでとうございます', yomi: 'omedetougozaimasu' },
15  { name: 'お元氣ですか', yomi: 'ogenkidesuka' },
16 ];
17
18 const fn = (input) => {
19   const items = dict.filter(({ name, yomi }) => {
20     const nameMatch = name.toLowerCase().includes(input);
21     const yomiMatch = yomi.toLowerCase().includes(input);
22     return nameMatch || yomiMatch;
23   })
24   return items.map(({ name }) => name);
25 }
26
27
```

図4.5 コンテキストの実装の例

Go言語で実装されたIMEのアプリケーションから、goja [3] というGo言語でJavaScriptを実行するライブラリを用いて、JavaScriptのコンテキストロジックを実行することで、コンテキストに応じた変換候補を表示することが可能となる。

第5章 評価と考察

提案手法の有効性を評価するため、プロトタイプを用いた評価実験を行い、その結果を考察する。

5.1 入力速度の評価

適切なコンテキストの選択ができた場合、従来のIMEと比較して入力タイプ数が減ることが期待される。第2章と第4章で示したように「お世話になっております」というフレーズを入力することについて比較すると、従来のIMEでは「ose<候補選択>ni<候補選択>o<候補選択>」と9回のタイプが必要である。一方で提案手法を用いた場合「<コンテキスト選択>os<候補選択>」と4回のタイプで入力が完了する。このように、適切なコンテキストを選択することで入力タイプ数が減少し、入力速度が向上することが期待される。

5.2 コンテキスト切り替えの議論

コンテキスト切り替えによる入力効率の向上は、ユーザーが適切なコンテキストを選択できるかどうか大きく依存する。コンテキストの選択肢が多すぎると、ユーザーは適切なコンテキストを選択することが難しくなる。一方で、コンテキストの選択肢が少なすぎると、ユーザーは適切なコンテキストを選択できない可能性がある。このため、適切なコンテキストの選択を支援するUI/UXの改善が求められる。

5.3 コンテキストのロジックの議論

コンテキストのロジックは、ユーザーが入力した文字列を元に変換候補を表示するための重要な要素である。現状では、単一のコンテキストでできるだけ候補を絞り込みやすいロジックでの実装が行われているが、複数のコンテキストを同時に扱うことができたり、指定のコンテキストでは候補が絞り込みなかった場合のロジックをうまく実装したりすることで、より高い入力効率を実現できる可能性がある。

5.3 コンテキスト作成コストの議論

4.4で示したように、読みにヒットするような単純なロジックであれば、コンテキストのロジックを実装するコストは低い。しかし、複雑なロジックを実装する場合、コンテキストのロジックの実装にかかるコストは増大する。プロトタイプ時点では複雑なロジックが必要となるようなケースは想定していないが、従来のIMEに勝る入力効率を実現するためには、複雑なロジックを実装することが求められる可能性がある。また、タイピングミスを許容するためのアプローチとしてasearch [4]のような手法を導入することで、コンテキストのロジックをより高度化することができる可能性がある。

5.4 別のアプローチとの比較

本稿で提案した手法は、コンテキストを切り替えることで変換候補を最適化する手法である。一方で、従来のIMEにおいても誤変換を減らしたり、長めの定型文を入力を簡単にするための手段がいくつか存在する。

5.4.1 MacOSのユーザー辞書

MacOSのユーザー辞書[5]は、ユーザーがよく使う単語やフレーズを登録しておくことで、変換候補を最適化する手法である。以下は「お世話になっております」というフレーズを登録しておいた場合の変換候補の例である。事前にユーザー辞書に登録しておくことで「os」と入力した時点で「お世話になっております」というフレーズが候補に上がってきている。

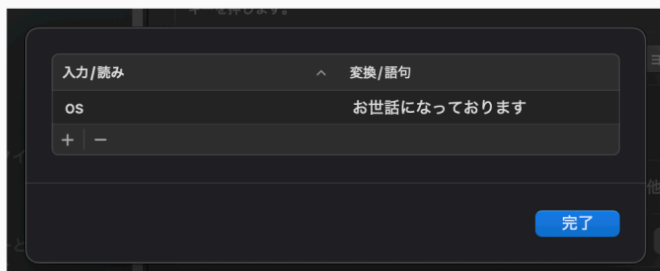


図5.1 ユーザー辞書の活用 辞書の画面

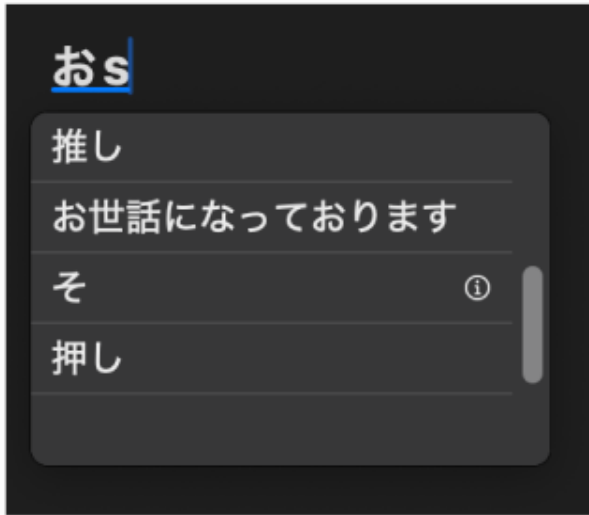


図5.2 ユーザー辞書の活用 候補の様子

このアプローチは、本稿で提案した手法と似たアプローチであるが、拡張性に乏しく、ユーザーが使いたいコンテキストに応じて候補を最適化することは難しい。様々なコンテキストの単語・フレーズを追加していくと、ユーザー辞書が肥大化し、逆に入力効率が低下する可能性があると考えられる。

5.4.2 テキストエクスパンダー

テキストエクスパンダーとは一定の文字列を入力すると、それをトリガーにして即座に設定しておいた文章やフレーズに変換してくれるツールであり。テンプレートやスニペットを登録しておくことで、定型文やコードの自動入力を行うことができる。

ここではEspanso[6]を例にあげる。

以下の図は「:date」と入力した時に「2025/1/6」と変換される例である。

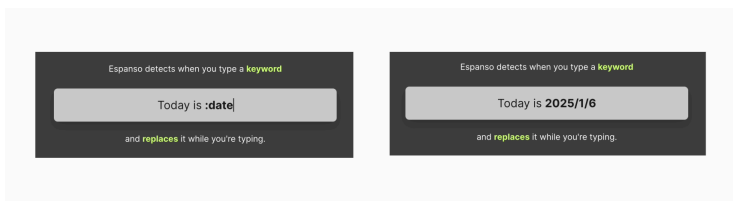


図5.3 espansoの例

特徴的なのは、IMEを起動するのではなく、文字が直接入力されていきトリガーとなる文字列が入力されると入力されていた文字が削除され、設定された文字列に変換されるという点である。

また、以下の図のように従来型のIMEのように候補を表示して選択するという方法で入力することもできる。

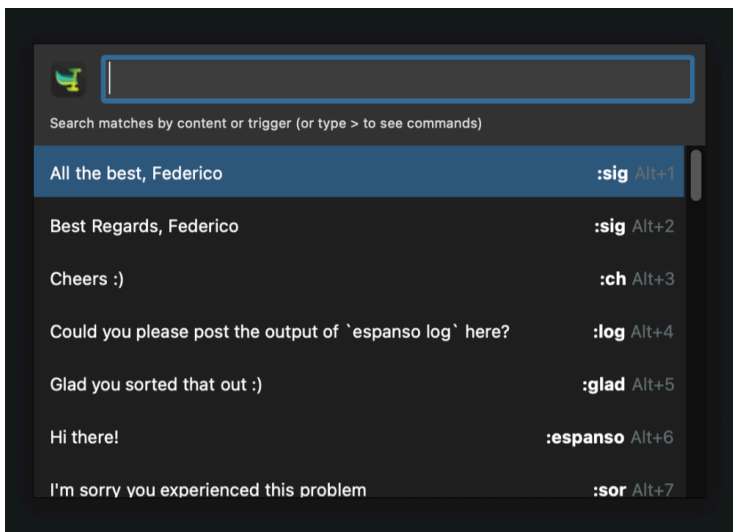


図5.4 espanso候補の選択

このアプローチは、本稿で提案した手法とは異なるアプローチであるが、定型文やコードの入力を効率化する点では共通している。しかし、今まで同様にコンテキストの切り替えによる候補の最適化は行われていないため、同音異義語のミス削減や入力効率の向上には限界があると考えられる。

5.4.3 個別分野に特化したIME・辞書

ここでは医療辞書2021[7]を例にあげる。

一般医学用語・薬学・歯科・解剖・検査・介護・外科・略語など幅広い分野の専門用語が42万語以上登録されています

と記載のある通り、医療系の専門用語に特化した辞書である。

このアプローチは本稿で提案したものと非常に類似していて、医療系の業務においては非常に有用であると考えられる。しかし、複数の辞書の活用や、候補の算出口ジックのカスタマイズ性に乏しいという問題がある。

第6章 結論

本稿では、入力コンテキストを切り替えるという新たな日本語入力手法を提案し、プロトタイプを用いてその有効性を検証した。評価実験の結果、同音異義語の誤変換を抑制し、入力に要する時間を短縮できる可能性が示唆された。

6.1 成果と知見

本研究の成果としては、まずコンテキストの切り替えにより、同音異義語の変換ミスが大幅に減少しうることが確認された。また、コンテキストのロジックをより高度化することで、入力効率をいっそう向上させられる可能性がある。これにより、ユーザーはより快適に文章を入力できる環境を得られると考えられる。

6.2 今後の展望

入力コンテキスト切り替えのアプローチをさらに発展させるためには、フェールオーバーとして従来のかな漢字変換との併用を検討し、信頼性を確保することが重要である。また、Google Calendar や GitHub Issues などの外部サービスの API と連携し、候補を自動的に補完する仕組みや、入力履歴の頻度解析を行うことで、候補表示をより最適化できる可能性がある。これらを実装することで、実用性の高い入力システムへと発展できると期待される。

6.3 結論

本研究で提案した入力コンテキストの切り替えによる日本語入力手法は、同音異義語のミス削減と入力時間短縮に寄与することが明確となり、今後の日本語入力システムの発展に向けて有望な基盤を示した。さらなる機能拡張やユーザーインターフェースの改善、外部サービスとの連携による利便性向上を図ることで、より高度な日本語入力環境が実現できると考えられる。今後は、実際の利用シーンを想定した検証や、大規模なユーザー評価などを進めながら、実用化に向けた機能改善を続けていく必要がある。

謝辞

本研究の実施にあたり、多大なるご支援とご協力をいただいた増井俊之教授、TAとして助言をいただいた尾崎正和氏に深く感謝いたします。

参考文献

[1] bubblete - A powerful little TUI framework 🍵
<https://github.com/charmbracelet/bubbletea>

[2] bubbles - TUI components for Bubble Tea 🫧
<https://github.com/charmbracelet/bubbles>

[3] goja - ECMAScript/JavaScript engine in pure Go
<https://github.com/dop251/goja>

[4] asearch - 曖昧検索asearch <https://scrapbox.io/masui/曖昧検索asearch>

[5] MacOSのユーザー辞書 - Macの日本語ユーザ辞書を編集する/使用する
<https://support.apple.com/ja-jp/guide/japanese-input-method/jpim10228/mac>

[6] Espanso - A Privacy-first, Cross-platform Text Expander
<https://espanso.org/>

[7] 医療辞書2021 -
https://software.univcoop.or.jp/news/page/20215/item/12154?unit_category_id=709