プロジェクト名：日本語ローマ字なぞり入力システムの開発
申請者名：高橋直希、三輪敬太

【提案プロジェクト詳細】

1 なにをつくるか

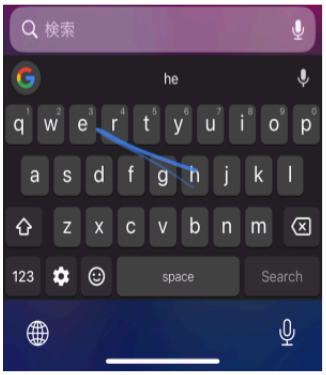
本プロジェクトで開発する日本語ローマ字なぞり入力システム「smoothie」は、なぞり入力のストロークからローマ字への変換機能とローマ字から日本語文字列へのかな漢字変換機能を備えた、史上初めての実用的な日本語なぞり入力である。このアプリは誰でも使えるようOSSで公開し、iOSアプリとしてもリリースする。

1.1 背景

1.1.1 なぞり入力について

なぞり入力とは、タッチディスプレイ上の仮想キーボードで、文字を指でなぞるように入力する方式のことを指す。ユーザーが単語を構成する文字の上を一筆書きのように入力すると、システムが入力パターンを解析し、意図した単語を推定する。これにより、キーを一つ一つタップする従来の入力方式と比較して、素早く直感的な入力が可能になる。

Gboard：英語



iOS標準：英語

iOS標準：中国語

1.1.2 日本語なぞり入力の現状と課題

英語をはじめとするローマ字アルファベットを書記に利用する言語圏や、キリル文字やギリシャ文字を利用する言語圏ではなぞり入力が広く普及し、一般的な入力方式の選択肢のひとつになっている。中国語はQwerty配列を利用した拼音漢字入力が行われるが、これもなぞり入力が普及している。しかし、日本語においては、そもそも実際に使える日本語のなぞり入力自体が存在しない。これは日本語の言語的な特性がなぞり入力の実現を阻んでいるためである。

具体的な課題として、以下の点が挙げられる。

1.1.2.1 日本語なぞり入力の課題

■ 分かち書きの欠如による入力単位の不確定性

英語では単語（word）ごとにスペースで区切られているため、単語を入力単位として変換システムを構築しやすく、なぞり入力でも1ストロークが1単語に対応する。一方で日本語は分かち書きをしないため、ユーザーが文字、音節、単語、文節、文など様々な単位で入力する可能性がある。この入力単位の不確定性が日本語なぞり入力の難しさを高めている。

■ ローマ字入力における母音の多用と**QWERTY**配列の不適合性

日本語をローマ字入力する際、母音（a, i, u, e, o）の使用頻度が非常に高い。一方、QWERTY配列では母音キーが固まって配置されているため、なぞり入力において密集している母音キーの中から入力したいキーを特定するという点で日本語入力には不向きだと言える。
さらに、日本語なぞり入力における入力補正の先行研究が不足しており、必要な知見が十分に蓄積されていないことも大きな課題である。
以上のような日本語特有の難しさにより、日本語なぞり入力の実現には、他言語と比べてより高度な言語処理技術が必要とされる。この点が、日本語なぞり入力の実用化が遅れている主な要因だと考えられる。

1.1.2.2 かな漢字変換の課題

■「ローマ字→かな→漢字」の2段階変換の必要性
日本語入力では、ローマ字からかなへの変換と、かなから漢字への変換という2段階の変換が必要となる。英語などでは無段階、中国語であってもローマ字→漢字の1段階の変換にとどまり、日本語のように2段階の変換が必要になるシステムは珍しい。

■ 誤字の補正を伴う変換処理
日本語なぞり入力では、ローマ字列に挿入/削除/置換を含む揺らぎが発生する。このような非決定的なローマ字列をナイーブに扱う場合、変換システムが考慮すべき可能性は大きく増加し、あり得る変換の探索範囲が広がりすぎてしまう。こうして変換システムが複雑かつ高負荷になることを避けるためには、アルゴリズム的な工夫が求められる。

■ 同音異義語の多さによる曖昧性
一般的な問題として、かな漢字変換自体がある程度困難なタスクであるという課題がある。日本語には同音異義語が非常に多く存在する。このため、音声だけから意図した単語を特定することが難しくなる。なぞり入力でローマ字から単語を推定する際にも、この同音異義語の問題が立ちはだかる。

1.2 日本語ローマ字なぞり入力システム smoothie

1.2.1 概要

これらの課題により、実用レベルの日本語なぞり入力システムを構築するには、多くの技術的ハードルが存在していた。

提案者はモバイルファーストなiOSキーボードアプリの開発経験があり、キーボードのUI設計からかな漢字変換エンジンまでの開発を経験し、アプリをリリースしている。この経験を活かし、モバイル環境に最適化された日本語なぞり入力システム「smoothie」を開発する。

smoothieは、UIからかな漢字変換エンジンまでを一貫して設計・開発することで、システム全体を俯瞰した最適化を行う。具体的には、なぞったストロークからローマ字への変換と、ローマ字からかな漢字への変換を緊密に連携させ、入力の揺らぎに強く、高精度な変換を実現する。

1.2.2 実装

なぞり入力のストロークからローマ字の変換を行い、ローマ字から日本語文字列を変換する。最終的にはストロークから日本語文字列を求めることができればよく、これを行うためには以下のように定式化できる。

ストロークzから最適な日本語文字列cを求める問題は、2段階の最適化問題に分割される。
R: 可能なローマ字列の集合, C: 可能な日本語文字列の集合 とする。
第1段階では、ストロークzから最適なローマ字列rを求める。

$$r^* = \arg\max_{r \in R}[p(r|z)] = \arg\max_{r \in R}[p(z|r)p(r)]$$

第2段階では、得られた最適なローマ字列r*から最適な日本語文字列cを求める。

$$c^* = \arg\max_{c \in C}[p(r|z)] = \arg\max_{c \in C}[p(r^*|c)p(c)]$$

以上の定式化から、日本語なぞり入力システムでは、ストローク解析モデル、ローマ字変換モデルらを組み合わせることで、最適な日本語文字列を求めることができる。本プロジェクトで開発するsmoothieは、これらのモデルを緊密に連携させることで、高精度ななぞり入力を実現する。

1.2.2.1 ストローク・ローマ字変換

ユーザーの入力したストロークを点列として扱い、ローマ字列への変換を行う。点列からローマ字列への変換では、ストロークの特徴量を基準とパターンマッチを併用することで行う。具体的には、以下の特徴量をストロークの解析で抽出し、その制約からストロークのパターンマッチすることでローマ字列を推定する。

1. 速度の極小点：ストロークの速度が極小となる点は、キー入力に対応していると考えられる。
2. 方向の変化点：ストロークの方向が大きく変化する点は、次のキーの入力を特徴づける重要な情報となる。
3. 曲率の極大点：ストロークの曲率が極大となる点は、キー入力に対応していると考えられる。

この特徴量による制約の下、動的に可能性のあるストロークのパターンを生成し、一致率を算出する。例えば極小点がn点であるとき、ストロークのパターンはアルファベット26文字のn乗あり、ここからローマ字列としてありえないものを除外してストロークを生成する。そして生成したストロークと入力のストロークを比較し、一致率からローマ字列ごとの確率を出力する。実際には極小点の周りのキーが入力されているものと仮定でき、極小点の周りのキーは最大でも7個であるため7^nのパターン程度の想定で良い。

全てをパターンマッチングすることによるアプローチも検討したが、有限パターンの単語でサポートできる英語と異なり、日本語では入力単位が一定ではないため、入力で想定されるローマ字列は無限長である。そのために無限長の文字列に対応するストロークのパターンマッチングが必要となる。これは計算量の面で現実的ではないと判断し、特徴量ベースの手法を基本とし、局所的にパターンマッチングを行う。プロトタイプでは音節単位のなぞり入力でパターンマッチを行い、高精度でローマ字列を推測できることを確認している。

特徴量の選択とそれぞれの重みの組み合わせについては、実験を通じて最適化を図る。最終的には、これらの特徴量を機械学習モデルに入力し、ストロークからローマ字列を出力するシステムを構築する。

以上のように、ストロークからローマ字列への変換では、ストロークの特徴量に着目し、その制約の中でのパターンマッチを行うアプローチを採用する。

1.2.2.2 かな漢字変換エンジン

かな漢字変換（またはローマ字列から日本語文字列への変換）には一般にかな漢字変換システムを利用する。このシステムの実装にはさまざまな方法があるが、近年の実装として単純かつ一般的な2-gramベースの手法を利用する予定である。モバイル端末上の仮想キーボードプロセスは通常のアプリケーションに加えて起動されるため、メインアプリケーションの動作を妨害しないよう、システムリソースがかなり制約されることになる。n-gramベースの手法はこうした状況下でも十分高速に動作し、かつ実用的な精度を達成する。提案者はすでにこの基本的なかな漢字変換エンジンを実装済みである。
一方で、提案プロジェクトにおいてはこの基本的なアルゴリズムにいくつか変更を加える必要がある。この大きな理由は、ユーザ入力であるストロークに対して、ローマ字列が一意に定められず、挿入/削除/置換を伴う揺らぎを持ったローマ字列が生成されるためである。このようなローマ字列に対して効率的なかな漢字変換システムを実現する必要がある。

従来システムでは入力文字列は単なる「文字列」として処理されるが、提案システムではこれを「文字グラフ」のような、より柔軟な構造に置き換えて取り扱う。このようなグラフベースのアルゴリズムは辞書引きの際の接頭辞検索や、2-gramに基づいた最適解検索のためのビタビサーチと相性が良いため、文字グラフを適切に構築できれば、処理速度を大きく悪化させずに揺らぎのあるローマ字列からの変換を実現できる。
実装のベースとしては、提案者がすでに開発を行っているAzooKeyKanaKanjiConverter[1]を基盤とする。このシステムはGitHubにおいてOSSで公開されており、実際にApp Storeで公開しているiOSキーボードアプリazooKeyや他のサードパーティのアプリでも利用されている。かな漢字変換の辞書データはTwitterやWikipediaから作成しており、こちらも同様にOSSで公開している。

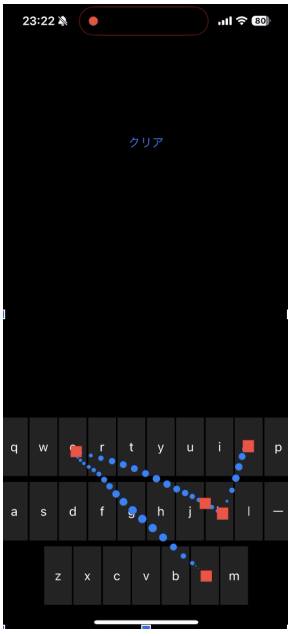
1.2.2.3 キャリブレーション機能

なぞり入力は直感的な入力方式ではあるが、特に日本語では他にないため慣れないユーザーにとっては戸惑いがある。また、ある程度の個人差が存在するほか、デバイスによっても振る舞いが変わる可能性がある。極端なケースでは、iPadとiPhone SEでそれぞれなぞり入力を行おうとした場合、指の動きは大きく異なることになる。

そこで、ユーザーがスムーズになぞり入力に習熟できるようにするため、チュートリアル機能を用意する。アプリ初回起動時に、簡単ななぞり入力の説明と練習用の単語を表示し、ユーザーがなぞり入力を体験できるようにする。さらに、チュートリアル中に行われるユーザーの練習入力を利用して、キャリブレーションを同時に行う。ユーザー毎の入力速度や切り返しの曲率の特徴をキャリブレーション入力から抽出し、ストローク解析アルゴリズムのパラメータを調整する。これにより、個々のユーザーに適応したなぞり入力の認識精度向上を図る。

このように、チュートリアルとキャリブレーション機能を組み合わせることで、ユーザーのなぞり入力習熟をサポートしつつ、個人差の問題にも対処することができる。ユーザーのなぞり入力に徐々に適応していくことで、使い続けるほど入力精度が向上していくことが期待できる。

1.3 プロトタイプ



プロジェクトの提案にあたり、実現可能性を探るためアプリキーボード上でなぞり入力をしたさいのストロークを表示・出力するアプリを作成した。どのような特徴量がiOS上で取得でき、それによってどのようなアプローチをするべきかの検討した。ストロークはjson形式で出力し、matplotlibでplotし分析も行った。

1.3.1 プロトタイプの詳細

iOSアプリとして作成した。上記の画像のようにテキストフィールドとキーボードのみが表示される。なぞることでそれぞれの時点における速さを青の丸のサイズで表現し、速度の極小点は赤の四角で表示される。

1.3.2 分析

1.3.2.1 パターンマッチングによる音節入力

音節単位ローマ字なぞり入力において50音(清音86文字，濁音25文字,)を単体で入力した時、それぞれのストロークを文字の直線のストロークとパターンマッチングを行い、想定された音節が入力ができるかを検証した。これによると合計111文字を計5回入力して誤検知は6回で約99%の精度で推論できた。

1.3.2.2 特徴量の分析

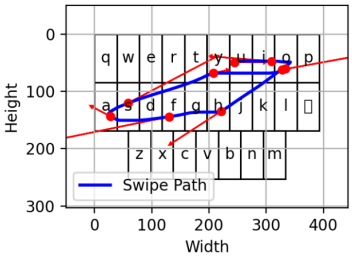
なぞり入力の先行研究として速度・加加速度[2]などを利用して入力文字を検出しているものもある。日本語なぞり

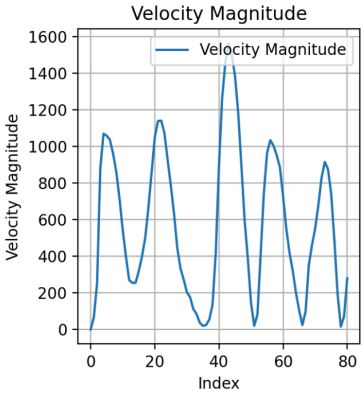
[1]AzooKeyKanaKanjConverter: https://github.com/ensan-hcl/AzooKeyKanaKanjiConverter/tree/develop
[2] “Modeling Gesture-Typing Movements” https://research.google/pubs/modeling-gesture-typing-movements/

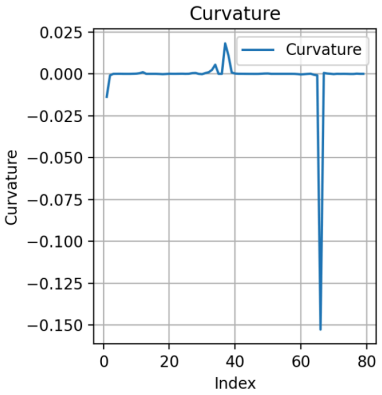
入力の入力検出の手法を検証している先行研究はなく、今回の分析では日本語なぞり入力を想定した時に速度の大きさ・曲率などの指標において入力を検出できる可能性があるか分析した。

下図は左から順番に、実際になぞったストロークの点列を表示したキーマップの図、ストロークの点列ごとの速度の大きさ、ストロークの点列ごとの曲率の大きさのグラフである。キーマップの赤い点はindexの10区切りに対応する。indexは以下の図は「ohayou」と入力した時ストロークと速度の大きさ・曲率のグラフである。それぞれのグラフを見ると入力したいキーの周辺で速度の大きさは落ち込み、特定の場合においては曲率が大きく変化する。他にもいくつかの日本語を入力した時の速度の大きさ・曲率の変化などを観察し同様の現象を確認している。

以上から日本語なぞり入力においてもこれらの値はなぞり入力のキー入力検出の特徴量として利用できると考える。







1.4 開発の目的と意義

smoothieの開発を通じて、以下の目的を達成する。

1.4.1 日本語入力の利便性向上

日本語ローマ字なぞり入力システムsmoothieの開発により、日本語入力の利便性を飛躍的に向上させることを目指す。高精度のなぞり入力からの日本語文字列への変換の実現により、ユーザーの入力速度の大幅な改善と、変換ミスに起因するストレスの軽減を図る。

1.4.2 日本語入力技術の発展への貢献

smoothieの開発を通じて得られた知見やノウハウを広く公開し、日本語入力技術全体の発展に寄与する。特に、オープンソース化によって、他の開発者や研究者が本システムを活用・改良できる環境を整備する。これにより、日本語なぞり入力の更なる高度化や新たな応用の促進を目指す。

2 どんな出し方を考えてるか

日本語入力は、利用するユーザがいなければ存在価値がない。このため、smoothieはiOSのアプリケーションとして実際にユーザに提供する。また、本プロジェクトから日本語なぞり入力を広く普及させるためにも、システムはオープンソースで公開し、無償で全ての開発者にも提供する。

3 斬新さの主張、期待される効果など

3.1. 斬新さの主張

本プロジェクトの斬新さは、日本語特有の課題に特化したなぞり入力システムの開発、ストローク解析とかな漢字変換の緊密な連携、そしてオープンソース化による日本語入力技術の発展への貢献の3点にあると考えられる。

■ 日本語特有の課題に特化したなぞり入力システムの開発
まず、英語や中国語では一般的なQWERTY配列でのなぞり入力を、日本語の言語的特性を考慮して最適化する点が挙げられる。具体的には、日本語入力に適したキー配置の工夫や日本語特有の課題に焦点を当てたアプローチを採用する。

■ ストローク解析とかな漢字変換の緊密な連携
次に、日本語入力における「ローマ字→かな→漢字」の2段階変換の問題に対し、ストローク解析とかな漢字変換を緊密に連携させることで、入力の揺らぎに強い高精度な変換を実現する点が挙げられる。この統合的なアプローチにより、日本語入力におけるユーザーの利便性を飛躍的に向上させる。

■ オープンソース化による日本語入力技術の発展への貢献
最後に、本プロジェクトで開発したシステムや得られた知見を、オープンソースとして広く公開する点が挙げられる。これにより、日本語入力技術の発展に寄与し、他の開発者や研究者がこれらを活用・改良できる環境を整備する。この取り組みは、日本語入力分野における日本語なぞり入力の技術の共有と発展を促進する点で革新的である。

3.2. 期待される効果

smoothieの開発により、モバイル端末での日本語入力革命と、なぞり入力による日本語入力の選択肢拡大という2つの効果が期待される。

3.2.1.モバイル端末での日本語入力革命

モバイル端末での日本語入力においてはフリック入力が広く利用され、QWERTY配列の利用者はやや少ない傾向にある。PC端末でQWERTYの利用経験を多く持つユーザにとって、新規にフリック入力を習得するメリットは小さい。一方で、モバイル端末においてQWERTYは仮名あたり打鍵数が大きい上に打鍵ミスが増えやすい配列であり、あまり効率的ではない。smoothieは、スマートフォンやタブレット端末でのなぞり入力に最適化されているため、より幅広い年齢層のユーザがより直感的かつ高速にテキストを入力できるようになる。これはより多様なバックグラウンドを持つユーザに対してより優れた入力方法を提供することになるため、モバイル端末を用いたコミュニケーションの円滑化や、情報発信・創作などの活性化も期待できる。

3.2.2.なぞり入力による日本語入力の選択肢拡大

現在主流のキーボードによる入力方式は、QWERTY配列をベースとしているため、必ずしも日本語の入力に最適化されているとは言えない。一方、smoothieのようななぞり入力方式は、入力動作がより自然で直感的であり、従来のローマ字入力に比べて素早くテキストを入力できる可能性がある。

なぞり入力は、日本語入力の選択肢を広げる革新的な技術だと言えるだろう。ユーザーは、自身のニーズや好みに応じて、キーボード入力となぞり入力を使い分けることができる。なぞり入力の利点を考慮すると、この新しい入力方式を選択するユーザーは増加していくことが予想される。

なぞり入力の普及が進めば、日本語入力はより快適で効率的なものへと進化していく可能性がある。この技術革新は、日本語でのコミュニケーションのあり方にも影響を与えるかもしれない。我々は、なぞり入力が日本語入力の新たなスマホ入力のスタンダードになりえると考える。

4 具体的な進め方と予算

4.1.開発

4.1.1.開発場所

各々の自宅

4.1.2.使用する計算機

- Macbook Air, M1 2020, macOS Sonoma 14.2.1
- Mac mini, M2 Pro 2023, macOS Sonoma 14.2.1

4.1.3.開発言語

iPhoneアプリ

- Swift

変換エンジン・学習

- Python
- Swift

4.1.4.使用ツール

Slack, Notion, Google Drive, GitHub, Git

4.1.5.作業分担

高橋	三輪
● iOSアプリの設計と開発 ● ストロークからのローマ字変換エンジンの設計と実装	● かな漢字変換エンジンの設計と実装 ● オープンソース化の準備と実行 ● ユーザーフィードバックの収集と分析

4.1.6.開発線表

月	内容	担当
6月	機能要件、非機能要件の洗い出しと仕様決定	三輪、高橋
	システム全体のアーキテクチャ設計	三輪、高橋
7月	ストローク取得・可視化ロジックの実装	高橋
	ストローク特徴量抽出ロジックの実装	高橋
	2-gramベースの言語モデルの実装	三輪
8月	ストローク解析モジュールとの連携	高橋
	特徴量マッチングによるローマ字候補生成ロジックの実装	高橋
	ローマ字列から日本語文字列への変換ロジックの実装	三輪

	プロトタイプを用いた動作検証と課題抽出	三輪、高橋
9月	プロトタイプで抽出された課題の解決	三輪、高橋
10月	ストローク解析モデルの組み込みとチューニング	高橋
	機械学習を用いたストローク解析モデルの開発	高橋
	言語モデルの高度化	三輪
11月	チュートリアル・キャリブレーション機能の実装	高橋
	ユーザーフィードバックに基づく変換改善	三輪
12月	単体テスト、結合テスト、システムテストの実施	三輪、高橋
	バグ修正と性能チューニング	三輪、高橋
	ユーザーテストの実施とフィードバックの反映	三輪、高橋
1月	App Storeへの申請準備	高橋
	ユーザーマニュアルや説明資料の作成	三輪、高橋
	ソースコードのコメント・リファクタリング	三輪、高橋
	成果報告の準備	三輪、高橋
2月	予期せぬ課題への対応バッファ	三輪、高橋

4.1.7.開発にかかわる時間帯と時間数

平均して一人当たり週19時間程度の活動を行う。日中は大学などの活動があるため、平日午後・土日を主な活動時間とする。月80時間/人 x 9ヶ月より1440時間の活動を予定する。

4.2予算

4.2.1.予算内訳

活動時間	活動時間 x 人件費
1440時間	2,880,000円

4.2.2.必要経費

特になし

5 提案者（たち）の腕前を証明できるもの

三輪

実績

日本語入力およびそのGUIを中心に、NLPやソフトウェアエンジニアリングの分野で実績がある。iOSの日本語入力キーボードアプリとして「azooKey」の開発を2020年9月に個人で開始し、Google日本語入力 / Mozcを参考に、変換エンジン、辞書データとその生成システム、GUIをそれぞれ独自に構築した。開発開始から3ヶ月で通常の変換や予測変換、曖昧入力などの機能を完成させ、App Storeで公開。その後過去3年半以上継続的に開発している。2023年2月にはソースコードをGitHubで公開し、OSS化を果たした。GUI面ではキーボード配列のカスタマイズの機能を搭載するなど、痒いところに手の届くアプリケーションとして徐々に知名度を高めており、日経産業新聞でも紹介された。App Storeでは公式キュレーションに入ったこともあり、執筆時点でApp Storeで平均評価4.6 (レビュー数160)と1万DLを達成した。GitHubでは200以上のスターを獲得している。変換エンジンは独立のライブラリとして公開しており、複数の第三者の製品でも利用されている[3]。また、iOSの国内最大規模のカンファレンスイベントであるiOS Developer Conference Japan 2023で登壇し、日本語入力開発の知見を発表した。

その他の活動

卒業研究 / 大関研究室（東京大学）
大学では卒業研究として「人間の視線データに基づいて言語モデルの振る舞いを評価する」という枠組みでの研究に取り組んだ。2024年3月の自然言語処理学会ではこの結果を元に、GPT-2の振る舞いをトークナイザの違いに関して評価した研究を発表した。
インターン / **Turing株式会社**
創業初期の2021年より2024年現在までインターン。初期の自動運転デモにおいて自動運転を担うAIの開発で中心的に貢献したほか、自動車の運転の制御、CANによる通信の解析、信号機認識AIの開発、車載ディスプレイのAndroidベースのOS開発およびJetpack Composeを用いたGUI開発など、ソフトウェア開発と機械学習を中心に開発に参加。共著だが成果の一部は論文にもなっている [4] [5]。
インターン / **Google Japan (Mozc・Google日本語入力・Gboardチーム)**
azooKeyの開発をきっかけとして、2022年より二度インターン。Mozcの開発に参加し、異なるフォント環境で変換候補を最適化する実装に取り組んだほか、辞書データの検証、変換の高速化、特殊文字の追加など様々な面で貢献。クローズドソースのため詳細は伏せるが、Android版のGboardの開発に参加。変体仮名のオープンソースフォント開発プロジェクトであるNoto Hentaiganaにも社内で貢献した。

高橋

実績

2019年アプリ甲子園においてARナビゲーションアプリ「HybridMap」で全国3位を受賞。韓国の釜山で行われた韓国科学アカデミー科学フェア2019で「剣道の自動審判システム」で優秀革新的研究賞受賞。3年次課題研究にて第63回日本学生科学賞入選1等受賞および第4回関東・甲信越静地区高校生探究学習発表会 ポスター日本語部門優秀賞。国際イノベーションコンテストにて3位入賞。2022年、企画開発をした「おしゃべりひろゆきメーカー」がACC TOKYO CREATIVITY AWARDSにて、総務大臣賞/ACCグランプリ、ACCシルバーを2部門で受賞。2023年Apple主催のWWDC Swift Student ChallengeでWinnerに選出。また、2024年にazooKeyの漢字変換エンジンを利用したOSSのApple Vision Pro用日本語エディタ「azooEditor」を開発しApp Storeでリリース。Apple Vision Proの日本語利用者より多くの喜びの声を受けている。またazooKeyの開発にも貢献した。

---

[3]「Forethumb」「推し活アプリ オシバナ(Oshibana)ウィジェット」「azooEditor」など
[4]SuperDriverAI: Towards Design and Implementation for End-to-End Learning-Based Autonomous Driving
[5] 自動運転のための大規模走行データセットを用いた深層学習による信号機認識

就業経験

株式会社LetterFan
2020年に著名人・アイドルとのビデオレターアプリ「LetterFan」のカメラアプリの実装および改善施策の実施
株式会社CoeFont
2020年に創業メンバーとして創業。株式会社CoeFontにてセールスおよびカスタマーサポート、サービス開発およびDX業務を行う。現在は同社のマーケティングとして「おしゃべりひろゆきメーカー」などの企画・開発を担当。
その他
詳細についてはNDAにより記載できないがiOSキーボードの開発などを受託し開発。

6 プロジェクト遂行にあたっての特記事項

三輪：未踏期間中に所属することになる研究室と勤務先の了解は得ている。
高橋：特になし

7 IT以外の勉強、特技、生活、趣味など

三輪

東京大学教養学部後期課程所属。言葉に広い関心を持ち、計算心理言語学を学んでいる。日本語全般に興味があり、日本語の用例収集が趣味である。最近の関心は「深掘る」「片思う」などの動詞が「深掘り」「片思い」などから新造される現象である。また日本の古字である「変体仮名」が好きで、出先で見かけた変体仮名の用例の写真を撮って集めている。書体やタイポグラフィにも関心があり、オリジナルの書体をデザインすることもある。

高橋

現在は早稲田大学基幹理工学部数学科に所属しており、集合論に興味がある。研究室配属などはまだだが、今後は集合論を中心として研究をしていきたいと考えている。また最近はマーケティングについて学んでおり、現在は効率的な広告運用やTikTokやYouTubeなどのShort Videoコンテンツの効率的なバイラルなどを個人的に試行錯誤して研究している。またプレゼンテーションなどが好きでWIPO Show and Tellプレゼンテーションコンテストなどにも開発したアプリのプレゼンなどを行い、1位獲得。

8 将来のITについて思うこと・期すること

三輪

現代社会には言語の壁を個々人の努力による研鑽で突破しようとする/させようとする風潮がある。しかし、このようなやり方には当然に限界があるし、膨大な時間が研鑽に注がれることによる損失は大きく、全体最適化を妨げている。近年著しく発展した自然言語処理技術をはじめとするIT技術に基づく仕組みとシステムによってこうした風潮を打破し、誰もが言語的なアクセシビリティの保障を享受する社会が実現することを強く期待している。

高橋

プログラミングスクールでの指導経験から、ChatGPTなどの補助ツールの登場にもかかわらず、多くの人にとって未だにプログラミングは難しいものである。プログラミングの最大の障壁は、自分の作りたいものを言語化し、それを詳細な要件にブレークダウンすることだと考える。将来的には、こうした障壁を下げるツールの発展により、より多くの人がプログラミングに取り組める世界が実現することを切に願っている。同年代の友人にもプログ

ラミングに興味を持ってもらい、共に学び合える環境が広がればと思う。