資料調査のための AI くずし字認識スマホアプリ「みを」

カラーヌワット・タリン（Google Brain）

北本朝展（ROIS-DS 人文学オープンデータ共同利用センター、国立情報学研究所）

概要：近年、くずし字認識は急速に進歩している。 KuroNet や Kaggle コンテスト優勝モデルなどは、くずし字データセット上で 90%を超える精度を達成している。これらのくずし字認識モデルを誰もが利用できるようにするために、我々は「みを」くずし字認識アプリを開発した。「みを」アプリはくずし字資料の写真に対してサーバでくずし字認識を行い、認識結果を画像上に表示することができる。「みを」アプリはくずし字認識モデルをプロダクションフェーズに持ち込んだと言える。本稿では、「みを」アプリの開発とリリースについて説明する。さらに、機械学習プロジェクトのライフサイクルと機械学習運用の観点から、くずし字認識研究の今後の方向性についても考察する。

キーワード：くずし字認識、機械学習、スマホアプリ、MLOps

“miwo” AI Kuzushiji Recognition Application for Document Examination

Tarin Clanuwat (Google Brain)
Asanobu Kitamoto (ROIS-DS Center for Open Data in the Humanities, National Institute of Informatics)

Abstract: Kuzushiji character recognition has made rapid progress in recent years. The recognition models such as KuroNet and the Kaggle competition winning models have achieved over 90% accuracy on the Kuzushiji dataset. In order to make these Kuzushiji recognition models available for anyone, we developed an application called “miwo” that can take a picture of the Kuzushiji document get the recognition result easily from Kuzushiji recognition system on CODH server. We can say that miwo app has taken Kuzushiji recognition models into the production phase. In this paper, we explain about the development and release of the miwo app. Then, we discuss the Kuzushiji recognition research from the perspective of machine learning project life cycle and machine learning operations. Finally, we propose ways to improve Kuzushiji recognition system in the future.

Keywords: Kuzushiji, Machine Learning, Smartphone App, MLOps

１．まえがき

機械学習研究の進展と大規模データセットの公開が契機となって、くずし字認識の研究は近年急速に進歩した。著者らが発表した KuroNet[1][2]、そして著者らが主催した Kaggle コンペ[3][4]の上位入賞者の認識モデルでは、学習データに対するくずし字認識の精度は 90%以上に達する。こうした認識モデルの一部はオープンソースとして公開されているが、くずし字認識のための計算基盤を整えて活用することは簡単ではない。そこで ROIS-DS 人文学オープンデータ共同利用センター（CODH）は、2019 年 11 月に IIIF Curation Platform を活用した KuroNet くずし字認識サービスを公開した。博物館や図書館は IIIF を用いて多くのくずし字資料を公開しているため、まず IIIF を対象としたくずし字認識システムを開発したのである。とはいえ、このサービスは IIIF で公開されていない資料に対応していないため、手持ちの資料をすぐに調査できないといった問題があった。そこで、くずし字認識システムに入力する画像データをスマホのカメラで撮影し、すぐにくずし字認識を行えるアプリがあると良いのではないかと考えた。そこで誰でも活用できるサービスとして開発したのが AI くずし字認識アプリ「みを」である。

機械学習プロジェクトの観点から見れば、KuroNet や Kaggle コンペを通してくずし字認識の概念実証（PoC）に成功し、「みを」アプリの公開によってくずし字認識はプロダクションフェーズに入ったと位置付けることができる。ただし、プロダクトをリリースしたことで、このプロジェクトが終わったわけではない。むしろ、この段階では、機械学習プロジェクトとしてはまだ半分しか終わっていない。多くの課題が残っているのである。そこで本論文は「みを」アプリ公開後の課題を、機械学習プロジェクトのライフサイクルという観点から考察するだけではなく、将来的にくずし字認識システムを改良する方向性を明らかにするために、データセントリック機械学習の考え方を紹介する。

図１：プロトタイプ版

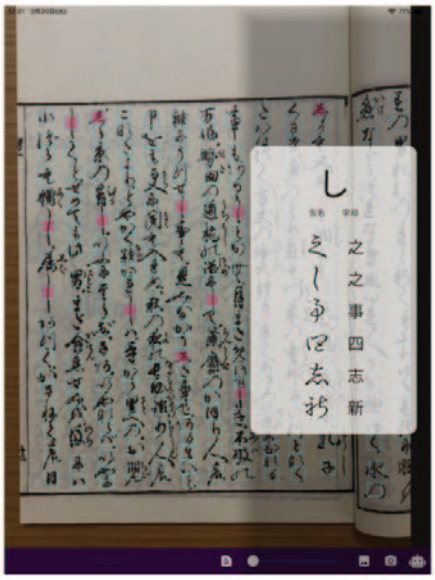

図２：KeMCo 版

図３：リリース版

２．くずし字認識アプリ「みを」

２．１ アプリ開発

「みを」は『源氏物語』第一四巻「みをつくし」にちなんだ名前である。「みをつくし」とは、「みを（船の水路）を示すために立ててある杭」の意である。「みをつくし」が人々の水先案内となるように、「みを」アプリはくずし字資料を読むための道案内となる、というのがアプリの基本コンセプトである。このようなコンセプトを実現するため、誰でも手軽に使えるよう、マニュアルを読まなくても使えるユーザインタフェースとし、ボタンを押す回数もできるだけ減らすようにした。また現在のハードウェア性能およびソフトウェア開発環境の制約を踏まえて、くずし認識はGPUを備えたサーバ側で行うこととした。スマホのカメラで資料を撮影すると、画像をサーバに送信し、くずし字認識を行い、認識結果を JSON 形式でアプリに返送し、文字を画像上に表示する仕組みを実装した。なおサーバ側では docker コンテナで動作する Flask アプリケーションでくずし字認識 API を稼働させており、Apache や nginx、gunicorn によるロードバランサを動作させることで負荷分散を行っている。

次にアプリの開発環境について述べる。スマホアプリを開発する場合、iOS と Android の 2 つの OS にどう対応するかが大きな問題となる。大きな開発チームを確保できる組織であれば、iOS 用に Swift、Android 用に Java か Kotlin などのネイティブ言語を用いて、2 つのアプリを並行して開発することも可能かもしれない。しかしこのような方法は、開発コストが高いだけでなく、運用開始後のメンテナンスも負荷が重く、例えば不具合が発見されれば両方のアプリを更新しなければならない。こうした問題を避けるために、「みを」の開発には Flutter[5]フレームワークを利用することにした。Flutter は Google から 2018 年 12 月に公開されたフレームワークであり、単一のコードベースで iOS と Android のアプリを同時に開発できる点に大きなメリットがある。「みを」を Flutter で開発したおかげで、2021 年 8 月 30 日の公開時には、iOS と Android の両方に対応しただけでなく、スマホとタブレットにも対応することができた。公開後のアプリダウンロード数を見る限り、iOS が 7 割、Android が 3 割である。どちらか一方の OS しか対応できなければ、ユーザの半分を失ってしまうことを考えると両方の OS に同時に対応できたことは有意義だったと言える。

くずし字認識モデルとしては 2 つのモデルが利用可能となっている。一つは KuroNet、もう一つは Kaggle コンペの優勝者である tascj チームのモデル[6]である。Kaggle コンペで用意したテストデータに対しては tascj モデルの方が KuroNet よりも精度が高いため、現在のバージョンの「みを」では tascj モデルを利用している。しかし「みを」のようにプロダクションで用いる機械学習モデルとして考えると、テストデータに対する精度が一番高いモデルが必ずしも最適とは言えない面がある。この問題については、第３．３節でさらに考察する。

２．２ 「みを」アプリの開発過程

「みを」の開発を開始してからリリース版まで、プロトタイプ版、慶應義塾大学ミュージアムコモンズ（KeMCo）で展示した体験版（以下 KeMCo 版）、そしてリリース版の 3 つのバージョンを開発した。以下では、アプリが 3 つのバージョンでどのように進化したかを振り返る。

２．２．１プロトタイプ版

プロトタイプ版は、アプリ開発が可能かどうか、どのような問題があるのかを大まかに把握するために開発した、いわば「みを」の MVP（Minimum Viable Product）である。MVP とは完璧な製品やサービスではなく、提供できる最小限のプロダクトのことを指す言葉である。「みを」の MVP として、資料をカメラで撮影し、画像をサーバに送信してくずし字認識を行い、その結果を受け取って画像上に認識結果を表示する、という最小限の機能を Flutter で実装することで、アプリが開発可能であることを確認した。またこのアプリの動作を撮影した動画を Twitter にアップロードしたところ大きな反響があり、こうしたアプリに関心を持つ潜在的なユーザが多いことも確認できた。

２．２．２ KeMCo 版

プロトタイプ版の動画がきっかけとなって、慶應義塾ミュージアム・コモンズ（KeMCo）の開館に合わせて開催される特別展覧会「交景 : クロス・スケープ」の特別プログラム「文字形－AI が開くくずし字の風景」にて、2021 年 4 月～6 月の間、新しい「みを」を開発して展示することになった。この展示のコンセプトについて KeMCo 関係者と共にブレインストーミングした結果、「みを」アプリに新しい機能を追加することが決まった。それが、認識結果の表示機能、変体仮名の字母表示機能、そしてテキスト出力機能である。

特にここでテキスト出力機能について触れておきたい。くずし字認識により文字がすでに読めているのだから、テキスト出力も簡単ではないかと思えるかもしれないが、実はここに著者らのくずし字認識と通常の OCR との違いがある。著者らのくずし字認識は文字とその座標を認識しているだけであり、その文字を並べて文字列として組み立てるにはまた別の処理が必要となるのである。特にくずし字資料は木版印刷や書写の自由度を反映してレイアウトが複雑なものが多いため、文字からテキストを組み立てるには文字の並びを柔軟に推定する手法が必要である。著者らはすでに 3 種類の手法[7]を試しているが、「みを」では Adaptive Rule-Based というルールベースの手法を採用した。機械学習ベースの手法である Deep Autoregressive Sequence model（Deep-AR）は精度としては低いわけではないが、GPU のリソースが必要となるだけではなく、ブラックボックスのモデルが意図しない間違いを出力することがあり、どこに間違いがあるかを確認するだけでも多大な手間を要することが分かった。「みを」アプリは機械のためではなく人間のためのアプリであるので、そこで間違いのパターンが納得しやすい Adaptive Rule-Based を採用することにした。

２．２．３リリース版

「みを」アプリ KeMCo 版に対して、慶應義塾大学ミュージアムコモンズの展示で体験した来館者からのフィードバックを整理し、これらをリリース版に反映させることとした。

まず「このアプリを古典文学の授業で使ってみたい」というコメントを検討した。もし「みを」アプリをくずし字学習に使うなら、学生にも興味を持ってもらえるように、アプリに学習支援機能を追加する必要があると考えた。くずし字を学習するためにもっとも大事な機能は何かといえば、それは読めない文字について調べられる機能だと考えた。そこで CODH が公開するくずし字データセットと連携し、くずし字の用例を検索する機能を追加した。

次にアプリの使い方を検討した。「みを」をどのようにユーザに使って欲しいのかについて、機械学習プロジェクトの企画という視点から考えてみると、おおむね 3 つの方向性が考えられる。すなわち、AI システムが人間の作業を支援する（AI Assist）のか、完全に自動化する（Full Automation）のか、あるいは半自動化する（Semi-Automation）のかという選択肢がある。全自動化ができれば人間にとって最も楽かもしれないが、現状のくずし字認識のレベルはそこからはるかに遠い。しかしプロトタイプ版と KeMCo 版には認識結果の修正機能がなかったため、認識結果が完全に正しくないにも関わらず全自動化システムに見えるという問題があることに気づいた。

「みを」が目指すのは全自動ではなく半自動である。そこで認識結果を修正する機能を「みを」に追加した。この機能は「AI が 100%正しいわけではない」こと、そしてくずし字認識の修正という作業には人間が関わる必要があることを明確に伝えるという役割も果たす。最後に、修正機能と共に、認識結果の保存機能も追加した。「みを」アプリはサーバ側にはデータを保存しないが、デバイス側では認識結果やユーザが入力した情報を Flutter 版の SQLite（SQFLite）に保存している。

このように KeMCo 展示からのフィードバックなども反映して機能を追加した結果、最終的に「みを」リリース版には、認識結果の修正機能、文字の検索機能、認識結果の保存機能などを追加した。これらの機能は、将来的には人間の作業を手伝うだけでなく、人間が学習して能力を向上させる機能にも繋げていきたいと考えている。また将来的には、機械学習をサーバで実行するだけでなく、オンデバイスでも実行できる機械学習モデルを用意することで、人間の作業時間を短縮する予定である。

２．３ アプリ公開

「みを」アプリは 2021 年 8 月 30 日に iOS と Android のアプリを同時公開した。2021 年 10 月末の段階で、ダウンロード回数は 33000 回以上、

またアプリを用いてくずし字認識した画像も 13 万件以上に達している。先述のように Flutter のおかげで iOS と Android を同時公開できたため、ダウンロード回数も増え、アプリ公開には大きな反響があった。

アプリの公開にあたっては、App Store(iOS)と Google Play(Android)の両方の規約を満たす必要がある。中でも重視されるのがユーザのプライバシーに関するポリシーである。「みを」アプリはユーザがサーバに画像データを送信するため、この画像をどう扱うかをプライバシーポリシーに明記しないとアプリ審査は通過できない。そこで「みを」では認識画像と認識結果をサーバに保存しないようにした。

アプリ公開後に判明した問題の一つに、アプリが動作しないデバイスの問題がある。もともと「みを」の開発では、多くのデバイスへの対応を重視していた。Flutter は iOS と Android の両方に対応してはいるものの、どちらかの OS あるいはデバイスに特有の不具合も生じるため、シミュレータではなく実物のデバイスでテストする必要がある。また Android のスマホに関しては、画面サイズが大きく異なるデバイスでのテストも必要である。このようにリリース前に多くのデバイスでテストを行ったが、いざリリースしてみると動作しないデバイスがあることが判明した。特に発売から 4-5 年が経過した古いデバイスでは問題が生じることが多く、OS のアップデートなどでは問題を解消できない。アプリのテストでは、最新のデバイスだけでなく、古いデバイスでもテストを行うことが望ましい。

3. 機械学習プロジェクトとしてのくずし字認識

3.1 くずし字認識というタスクの特殊性

くずし字認識プロジェクトは、機械学習用データセットの公開から始まり、機械学習モデルの開発、コンペの開催などを経て、最終的にはアプリの公開にまでこぎつけることができた。またその過程では、機械学習のプロジェクトとしては珍しいほどの大きな反響があった。そこで以下ではくずし字認識プロジェクトの全体像を振り返るとともに、このプロジェクトの特殊性や展開などについて議論する。

最初に考えておきたいのが、くずし字認識というタスクの特殊性である。今回の「みを」アプリの公開に対して、SNS では大きな反響があった。アプリに関するコメントは何千件も寄せられた、多くのコメントはアプリが実際にくずし字を読めることを絶賛していたが、それに対して「アプリは全然文字を読めていないではないか」という否定的なコメントも見られた。なぜこのように評価が大きく割れてしまうのか。その原因の一つに、くずし字認識というタスクの特殊性がある。

そもそも機械学習モデルの推定結果が使えるかどうかは、ユーザ側が期待するパフォーマンスに依存して相対的に決まるものである。中でもわかりやすいのは、人間のパフォーマンスを機械学習が上回るかどうかという基準であり、画像認識タスクで AI の分類精度が人間の分類精度を上回ったことがニュースになるなど、機械学習の有効性の宣伝文句にもこの基準がよく使われる。しかしくずし字認識タスクにこの基準を単純に当てはめようとすると問題が生じる。「人間のパフォーマンス」の分散が非常に大きいというのが、くずし字認識タスクの特殊性だからである。

まずくずし字が読めない大半の現代日本人(人口の 99.99%)にとって、人間のパフォーマンスは 0%に近いか十分に低いことから、くずし字認識に多少の間違いがあっても AI は既に人間レベルを軽々と超えている。ゆえに、アプリはすごいという絶賛の声があふれることになる。一方、くずし字がきちんと読める人（人口の 0.01%）にとって、人間のパフォーマンスは 100%に近いことから、現状のくずし字認識にはミスが多いという評価になり、否定的なコメントをすることになる。このようにどのレベルの人間のパフォーマンスを基準にするかによって、アプリの評価は全く逆転してしまうのである。

専門知識が必要なタスクにおいては、タスクに関する教育を受けた人と受けない人のパフォーマンスの差が大きくなるのは一般的な傾向である。例えば医療画像分類のタスクでも同様の状況は生じるだろう。専門知識が必要なタスクのユーザは多くの場合は専門家であり、専門家と一般の人々とのパフォーマンスの差が問題になることは少ない。ところがくずし字認識では、専門家と一般の人々とのパフォーマンスに大きな差があるにもかかわらず、ユーザの主力は一般の人々なのである。これがくずし字認識というタスクの特殊性であり、アプリがやや過大気味な評価を受けるマジックでもある。このような傾向を示す機械学習タスクは珍しいかもしれない。逆に言えば、多数の人々が AI の恩恵を受ける可能性を秘めているという点で、くずし字認識タスクの重要性を示しているとも言える。

なお、専門家と比べるとくずし字認識のパフォーマンスが低いとしても、それはくずし字認識が無用であることを意味するわけではない。というのもくずし字認識のスピードに関しては、機械は人間よりもはるかに優れているからである。いくら人間が確実に読めるとは言っても、それは時間をかければ読めるということであって、画像を見てパッと内容を把握できるわけではない。現代日本人である以上、やはり現代の日本語文字を見る方が読むスピードは圧倒的に早くなる。その意味では、資料の内容をざっと把握する、あるいは翻刻の下読みをするという目的であれば、くずし字認識は専門家にとっても有用なツールなのである。我々が「みを」を現場での資料調査ツールと位置付けているのは、この目的での有用性を踏まえたものである。

3.2 機械学習プロジェクトのライフサイクル

機械学習プロジェクトを運営するには、プロジェクトのライフサイクルを理解する必要がある。それはもし予想外の問題が起きたら、どの段階で何を改良しなければならないのかというヒントが得られるからである。機械学習プロジェクトのライフサイクルにはさまざまなパター

ンがあり、詳細はプロジェクトごとに異なるが、くずし字認識プロジェクトはは図４[8]のようなサイクルに近い。

図４のライフサイクルでは、まず問題定義を行い、その問題を解くためのデータセットを作成する。次にデータセットに対して機械学習を適用し、パフォーマンスが優れた学習モデルを作成する。そしてモデルを実環境に展開して運用がスタートする。ここまでは機械学習プロジェクトとしてイメージしやすい部分である。

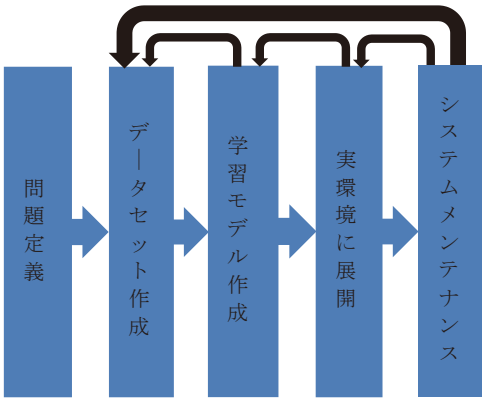

図４:機械学習プロジェクトライフサイクル

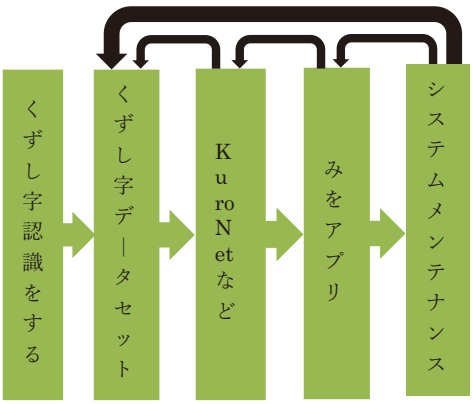

図５:「みを」プロジェクトライフサイクル

しかし実際には、実環境に展開した後に、システムメンテナンスしながらエラー分析を行い、システムが実環境でうまく稼働するかを確認する必要がある。もしシステムがうまく稼働しない場合、どこが問題なのかを探さなければならない。問題はアプリやサービスの不具合かもしれないし、学習モデルかもしれない。あるいはデータセットにまで遡らないといけないかもしれない。この部分も機械学習プロジェクトを実環境で活用するために欠かせない部分である。つまり、アプリやサービスを実環境に公開することが、機械学習プロジェクトの完了を意味するわけではない。むしろ公開をもって、ようやくプロジェクトの半ばにたどり着くと考えた方がよい。「みを」アプリの場合も、アプリの公開はライフサイクルの半分に過ぎない。多くのユーザがアプリを使用するおかげで、システムを監視することが可能になった。今後は、問題を見つけ、どの段階の改良が必要かを見極めることが課題となる。

３．３ PoC（概念実証）から運用へ

一般的な機械学習研究は、ベンチマークとして使うデータセットをダウンロードし、できるだけ認識精度が高い学習モデルを作成した後、そのモデルの内容を研究論文として投稿するという流れで進む。このように学習モデルを作成するまでのフェーズを PoC（Proof of Concept、概念実証）と呼ぶ。このフェーズでは、プロジェクトの概念やアイデアが実現可能であることを証明できればよく、KuroNet や Kaggle モデルもくずし字認識プロジェクトの PoC と位置付けることができる。しかし、ここから運用フェーズに入ると、学習モデルの精度以外にも検討すべき事項が多数出てくる。PoC は機械学習プロジェクト全体の 5%に過ぎないとも言われている[9]。

PoC では成功したように見えても、実環境ではうまくいかないプロジェクトは少なくない。その一つの原因は、PoC と運用におけるデータセットとモデルの関係の変化という点にある。PoC では、データセットを固定し、モデルを変える。しかし運用に入ると、モデルは固定され、入力するデータの方が変わる。もし運用時に入力されるデータが、PoC で用いたデータから変化してしまうと、PoC の段階でいくら精度が高くても運用の段階では精度が上がらないことになる。

ここで、データの変化について特に考慮すべきなのが、Data Drift と Concept Drift の問題である。Data Drift とは入力データの変化を指し、認識モデルが x → y で学習したのに、実環境で z → y に変化してしまう場合に対応する。一方、Concept drift とは入力データと出力データの関係の変化を指し、認識モデルが x → y で学習したのに、実環境では x → z に変化してしまう場合に対応する。いずれの場合も、学習モデルはうまく認識できなくなり、パフォーマンスは低下してしまう。「みを」アプリも Data Drift と Concept Drift の問題を抱えている。

まず「みを」アプリにおける Data Drift とは、ユーザが認識したいデータとモデルが学習したデータが異なるという問題を指す。くずし字認識モデルが学習したくずし字データセットは江戸時代の印刷された版本が中心なのに対し、「みを」アプリのユーザは古文書や掛け軸、博物館の展示物などの写本が多いため、文字の字形や出現頻度が大きく異なる。KuroNet はそうしたデータをあまり学習していないため、精度は大きく低下してしまう。また画像の撮影環境にも違いがある。くずし字データセットの画像は研究用としてきちんと公開するために、撮影環境（明るさや背景の色）を注意深く調整しており、画質も高いものが多い。一方、「みを」アプリで撮影された画像はデバイスごとに画質が様々であり、博物館のように照明が暗い場所で撮影されることも多い。こうした Data Drift を避けるためには、アプリで撮影した画像の明るさを認識前に調整したり、データセットまで遡って写本のデータを増加させたりするなどの対応が必要かもしれない。

次に「みを」アプリにおける Concept Drift は、江戸時代の文字で学習し、平安時代の文字を認識

する場合などに対応する。平安時代の変体仮名の字母は江戸時代より多く、当時はひらがなとして使われていたが、江戸時代になると漢字としてしか使われない文字が少なくない。このような場合、ある文字を漢字と判定するか、ひらがなと判定するか、時代によって正解が変化することになる。こうした問題は博物館の古い資料を認識する場合などに発生する可能性があるが、こちらで対応できる面も少ないため、現段階では Concept Drift より Data Drift の方を優先して対応する予定である。

3.4 エラー分析

一般的に機械学習プロジェクトでは、PoC フェーズにおいてモデルを改良する過程では徹底的にエラー分析を行い、エラー分析からモデルを改良する指針を得ることが望ましい。くずし字認識プロジェクトにおいても、どのような文字、どのような入力画像などでエラーが発生しやすいかを把握した上で、モデルのアルゴリズムを改良するか、データセットを改良するかなど、対応の優先順位を決める必要がある。データセットの改良とは、データを増やすことだけではない。データセットに存在する問題の改良、例えば曖昧なラベルを除去したり、ラベルの付与に一貫性を持たせたりすることも重要である。この問題は第 4 節でさらに詳しく説明する。

くずし字認識プロジェクトの場合、変体仮名に関する曖昧なラベル、例えば「見」や「屋」など字母になりうる文字の問題がある。これらの文字のラベルを漢字にするか、ひらがなにするかは、人間が翻刻する段階で文脈を考慮して決定することになっている。ところが機械は文字の字形だけを見て判断しているため、ほとんど同じ文字に 2 種類の異なるラベルが付与されているように見える。その結果、モデルがどちらのラベルを出力するかは、データセットにどちらの文字が多いかで決まってしまうことになる。この問題を解決するには、人間と同様に機械も文脈を見る必要があるため、言語モデルとの統合という可能性も研究課題となる。ただし近世以前の日本語についてはコーパスが十分に揃っていないため、言語モデルの構築は簡単ではないという問題が残っている。

くずし字認識プロジェクトのもう一つの問題は、漢字の認識精度が低いという問題である。画像認識プロジェクトの精度向上にはデータセットの大規模化が寄与している現状を踏まえれば、くずし字認識プロジェクトでもデータセットの拡大が望ましいことは明らかである。しかしくずし字データセットの作成はコストが高いため、やみくもにデータセットの拡大に走ることが正解なのかは十分に検討する必要がある。例えばエラー分析によって認識精度が低い漢字を特定し、そうした文字に絞ってデータセットを拡大すれば、より効率的にデータセットの大規模化が実現できる可能性もある。

3.5 監視

システムの監視は運用で生じる問題を特定するために重要である。監視すべき Metrics には、Software Metric, Input Metric, Output Metric の 3 種類がある。以下では「みを」の監視に関する課題をまとめる。

まず Software Metric は、ソフトウェアに関する指標である。アプリ側ではメモリ使用率や認識にかかる遅延、またサーバ側では GPU の使用状況や負荷、API に生じたエラーなどである。サーバ側では監視ツールとして Sentry[10]というツールを導入した。このツールで API のエラーを分析することにより、サーバに送信された画像の 0.4% 程度にエラーが生じていることをつかんだ。さらにエラーを詳細に分析すると送信データのサイズが 0 バイトになっているため、アプリ側の画像の扱いに問題が生じている可能性がある。おそらくこれは、古いデバイスに関する何らかの互換性が問題となっているのであろう。あるいは、Flutter の使っている画像入力ライブラリのバグも考えられる。

次に Input Metric は、入力データに関する指標である。現在の「みを」はプライバシー保護の観点から入力データを保存していないため、入力データに関する詳細な分析はできていない。しかし今後はプライバシーを含まない情報として、デバイスに関する情報、画像の平均の明るさ、画像のサイズ、認識に要した時間などを監視し、「みを」の改良に役立てたいと考えている。さらにエラー情報を積極的にフィードバックしてもらう機能などをつければ、監視機能をさらに向上させることができる。

最後に Output Metric は、出力データに関する指標である。こちらについてもプライバシー保護の観点から現在は認識結果を保存していないが、認識した内容に関係のない認識結果の文字数などは監視対象にできるのではないかと考えている。もし何も認識できない結果が多ければ、アプリに不具合が生じている可能性もある。また KuroNet や Kaggle モデルは、画像中の文字数に応じて認識時間は長くなるが、文字数が少ないのに認識時間が長い場合、アプリかサーバに不具合が生じている可能性がある。またこちらについてもユーザからのフィードバックは重要である。認識誤りを編集した結果をユーザから積極的に収集する機能を追加し、ユーザが許可した範囲で情報を収集しデータセットの強化に活用したいと考えている。

プライバシー保護を優先しつつそれとは関係のない指標を収集すること、またオプトイン方式でユーザからのフィードバックを積極的に収集すること、この 2 つの方向性がアプリとモデルを改良しサービスを向上させるための監視として重要だと考えている。

4．モデルセントリック AI からデータセントリック AI へ

3．3 節で述べたように、機械学習の研究はデータセットをダウンロードし、モデルを作成し、アルゴリズムの改良でパフォーマンスを向上させるのが一般的である。このような方法はモデルセントリック（モデル中心）AI ともいう。これ自体に問題があるわけではなく、そもそもモデルがないと何の問題も解決できないのも確かである。しかし、機械学習をうまく実環境で運用するには、

それだけでは足りない。

もし、ある程度は精度が高いモデルがすでにあるとして、AI システムのパフォーマンスを向上させる方法は、アルゴリズムの改良しかないのだろうか。AI はデータとアルゴリズムで作られるとも言われる。モデルセントリック AI はアルゴリズムに注目するものの、同じくらい大事な要素であるデータにはほとんど注目しない。しかし、前述したように、AI プロジェクトがプロダクションフェーズに入ると、モデルが固定され、データが変化するため、モデルの精度を向上させても問題は半分しか解決しなくなる。近年の機械学習研究では、このような問題点に注目して研究するデータセントリック AI [11]への注目が高まりつつある。データセントリック AI コンペや、NeurIPS のような大規模学会でのワークショップも開催されるようになってきた。

データセントリック AI は運用フェーズとも似ており、モデルを固定し、データの量と質で AI のパフォーマンスを向上させる。データの質をどのように改良するのがいいかは、問題によって異なる。大抵の場合は、データを増やすことと、曖昧なデータラベルを減らすことが重要である。

まずデータ増加については、何も考えずにデータを増やすのがよいわけではない。質の悪いデータを追加すると、AI の結果も悪くなる。エラー分析に基づき、モデルの弱点を改良する可能性があるデータを増やすことがまず大事である。なぜなら、データの作成にはコストがかかるからである。徹底的なエラー分析から始めることで、できるだけ低いコストで大きくパフォーマンスを改良することが可能となる。

図６：良いデータ拡張

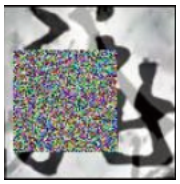

図７：難しすぎるデータ拡張

データ増加は、新しいデータを追加することが可能であればそれが最も良いが、もしそれが不可能であれば、データ拡張も検討すべきである。ただし注意すべきは、難しすぎるデータ拡張は曖昧なラベルを生み出し、モデルのパフォーマンスを低下させる恐れがあるという点である。データが難しすぎるかどうかに関する一つの判断基準は、人間がそのデータを見てもまだ区別できるかを見るというものがある。人間のパフォーマンスならばまだ正解がわかるというレベルが望ましい。

一方、曖昧なデータラベルの処理については３．４節で説明したこと以外に、データを作成する段階でラベル付与のルールを明確に決めることも重要である。くずし字データセットの場合、Kaggle コンペのデータセットを作成する前は旧字体と新字体のラベルがバラバラだったため、こちらで両方の文字を統一して新たなデータセットを作成した。曖昧なラベルの処理は、実はくずし字認識研究の大きな課題である。今後、数千人の参加者がいる「みんなで翻刻」[12]プロジェクトのデータを学習データとして利用する際には、バラバラのラベルを統一する方法を考える必要がある。

データセントリック AI はくずし字認識研究に不可欠の考え方である。例えばくずし字データセットの範囲を考えてみよう。くずし字データセットは特定の時代、特定の古典籍の種類から作られたものである。そのため、実世界に存在するすべてのくずし字資料から見ると、非常に小さい範囲しかカバーしていない。たとえ、くずし字データセットでモデルに学習させて精度が 99%に辿り着いたとしても、データセット外の資料をうまく認識することができるとは限らない。KuroNet や Kaggle モデルはくずし字データセットに対してはある程度精度が高いため、今後の課題はアルゴリズムの改良よりも、むしろデータセットの外へ認識可能な範囲を拡大することにある。これからのくずし字認識研究は、モデルセントリックからデータセントリックへと変わっていくべきだと考えている。

５．まとめ

「みを」アプリの公開によって、くずし字認識プロジェクトはプロダクションフェーズに入った。KuroNet や Kaggle に取り組むフェーズではアルゴリズムが焦点であり、さまざまなパラメータを調整することで精度を上げることに関心が向いていた。しかしプロダクションに入るとアルゴリズムは固定されるため、むしろデータが中心的な存在となる。そこで問題となるのが、PoC で開発された機械学習モデルが実環境でもそのまま使えることは少ないという問題である。実環境では、コントロールできない要素が入ってくるため、失敗の原因を分析することも簡単ではない。またデータセットでは学習していないタイプのデータが入力され、それに対しては正解ラベルもないため、精度の面で低下を免れない。これらの問題に対応するには、システムを監視し、徹底的なエラー分析を行って、原因を探し出すことが重要である。またサービスの課題も多々あるため、優先順位を決め、計画的に改良を進める必要がある。このように「みを」の公開をきっかけとして、従来は検討していなかった新たな課題が見えてきた。

「みを」アプリは多くのユーザにくずし字認識モデルを提供し、くずし字資料をより簡単に調査する可能性を広げた。今後は古典文学や歴史の研究を支援するツールの一つとして成長させていきたい。

参考文献

[1] Alex Lamb, Tarin Clanuwat, Asanobu Kitamoto (2020) “KuroNet: Regularized Residual U-Nets for End-to-end Kuzushiji Character Recognition” Springer Nature Computer Science Special Issue on Document Analysis and Recognition
[2] Tarin Clanuwat, Alex Lamb, Asanobu Kitamoto (2019) “KuroNet: Pre-Modern Japanese Kuzushiji Character Recognition with Deep Learning” The International Conference on Document Analysis and Recognition (ICDAR)
[3] 北本 朝展, カラーヌワット・タリン, ボーバー・イリザー ミケル（2020）「Kaggle くずし字認識―世界規模の人文系コンペ開催への挑戦―」、人工知能学会誌 35(3)、366 - 376
[4] Kaggle Kuzushiji Recognition, https://www.kaggle.com/c/kuzushiji-recognition
[5] Flutter Framework https://flutter.dev/
[6] tascj チームのくずし字認識モデル https://github.com/tascj/kaggle-kuzushiji-recognition
[7] Alex Lamb, Tarin Clanuwat, Siyu Han, Mikel Bober-Irizar, Asanobu Kitamoto (2021) “Predicting the Ordering of Characters in Japanese Historical Documents” (arXiv)
[8] Machine Learning Engineering for Production (MLOps), Coursera.
[9] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, Dan Dennison (2015) “Hidden Technical Debt in Machine Learning Systems” (NIPS)
[10] Sentry, Application Monitoring tool, http://www.sentry.io
[11] A Chat with Andrew on MLOps: From Model-centric to Data-centric AI, https://www.youtube.com/watch?v=06-AZXmwHjo&t=69s
[12] みんなで翻刻プロジェクト https://honkoku.org/

図８：リリース版のテキスト出力画面

図９：リリース版の文字修正画面

図１０：リリース版の保存された認識結果のリスト

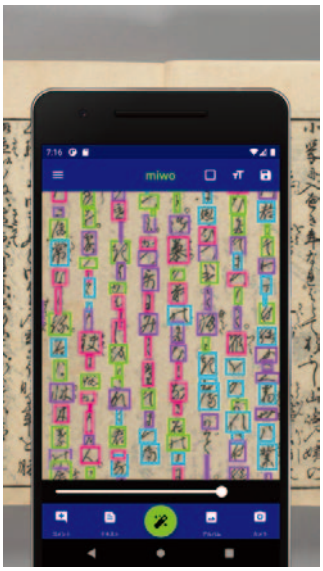

図１１：リリース版のバウンディングボックス表示画面