

AI-OCR機能の実装

酒井辰則

開発における問題点

AI-OCR製品は数多く販売されているが、クラウド利用が前提であったり、継続的な学習の可否、カスタマイズによる機能拡張性、AIを適用する範囲など、製品によって特徴が様々なため、利用者側で想定する全要件を満たす製品を選定するのは難しい。

製品の選定にあたっては、必ず必要な機能を持つ製品の中から選ぶ必要があり、選択肢が限られてしまう。

手法・ツールの適用による解決

昨今AI技術の発展により、ChatGPT・Copilotなどに指示を出すことで、ノーコード・ローコード開発が可能となってきている現状がある。

そこで本修了制作では、過去に経験したAI-OCR導入プロジェクトで、製品の選択肢を狭める要因となったいくつかの要件(特に帳票識別機能)について、自作でどこまで実装が可能か検証を行った。



過去のAI-OCR導入プロジェクトでの気づき

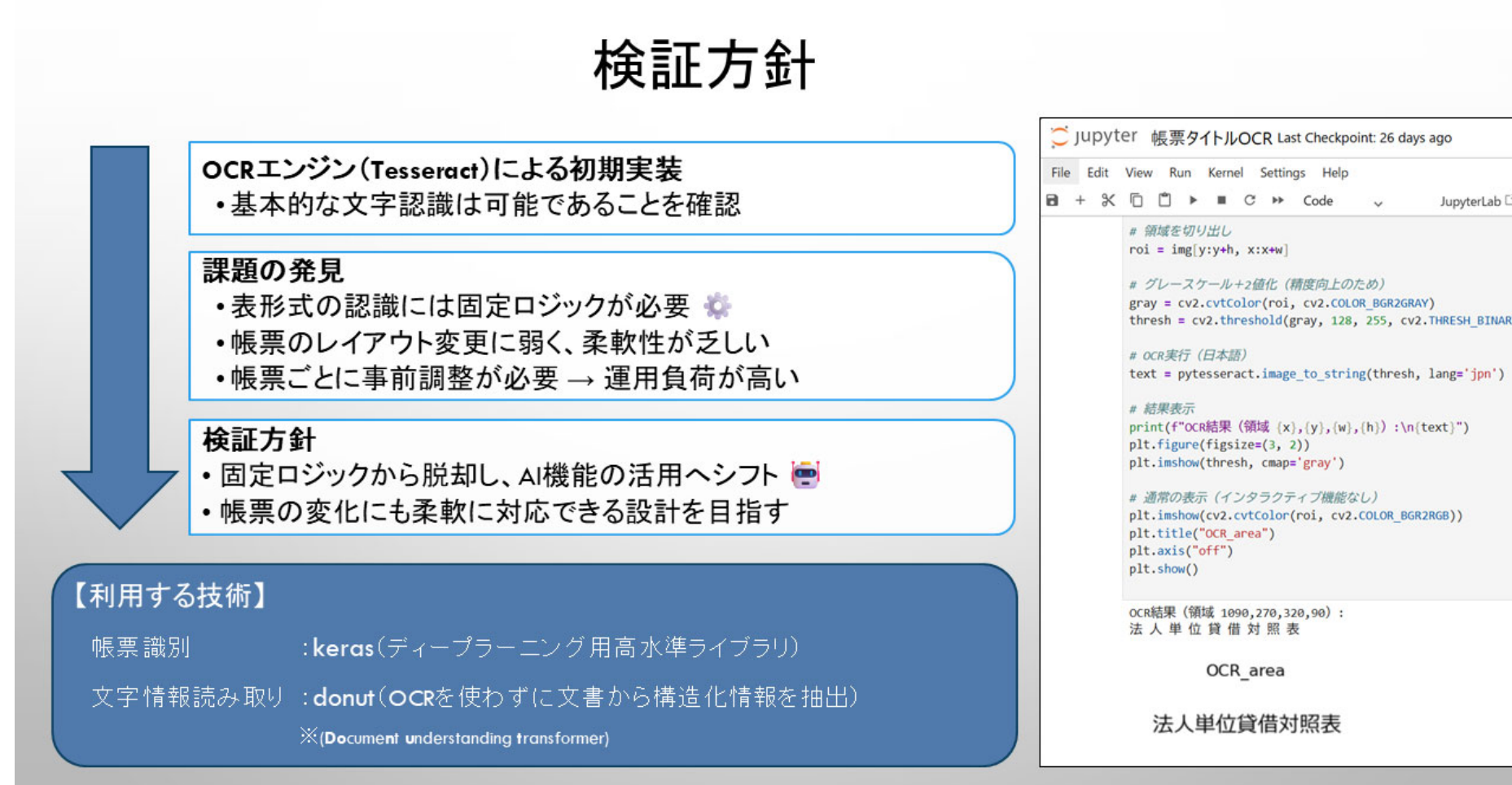
- 市販のAI-OCR製品は、クラウド依存・学習可否・カスタマイズ性などが製品ごとに異なり、全要件を満たす選定は困難。
- 内製は人的リソースの制約から減少傾向にあるが、近年のAI技術進展により、ノーコード・ローコード開発が現実的に。
- 本制作では、AI-OCR導入時の要件が自作でどこまで構成可能かを検証した。

修了制作で検証したAI-OCRの機能

過去実際に携わったAI-OCR導入プロジェクトでの要件について、自作による実現可能性を検証。

- ①インターネット非接続環境での動作(セキュリティ観点)
 - ローカルPC上での動作を実現
- ②継続的な学習
 - 学習結果のモデルをローカル環境に保存、再利用を可能とした
- ③文字情報読み取り、帳票識別(※)
 - 2種類の帳票の分類に成功。文字認識については精度が出なかった

※いずれも読み取る対象はテキストデータを持ったPDFではなく、PNGファイル。



①インターネット非接続環境での動作

セキュリティ観点から、読み取り対象のデータをインターネット上に上げないという要件

外部サービスを利用せず、ローカルPCにインストールしたJUPYTER NOTEBOOK上で帳票分類、画像からのテキスト抽出の確認ができたことから実現可能。

これにより、学習結果のモデルをローカル環境にファイルとして保存することが可能。



②継続的な学習

学習を継続することでどんどん賢くなってほしいという要件。

AI-OCRに対して期待する継続的な学習は以下を想定。

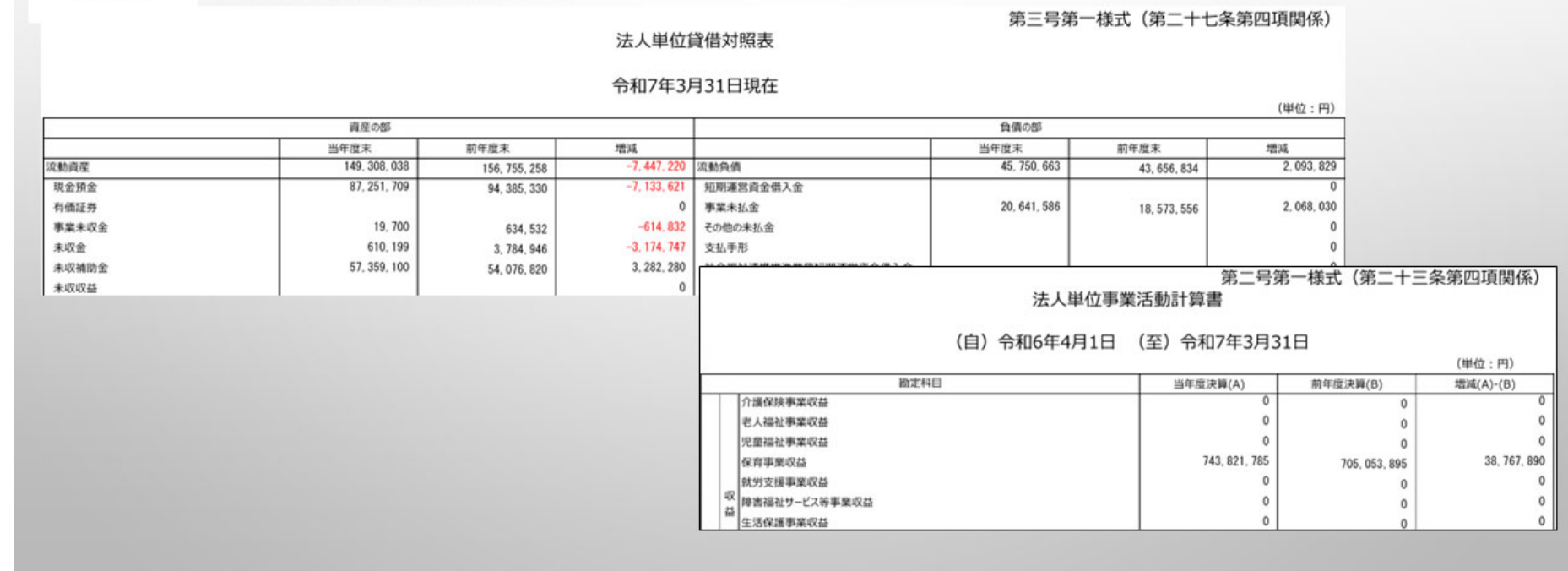
- ・文字情報読み取りに関する学習
- ・帳票の識別、分類に関する学習

ローカル環境で実行でき、学習結果のモデルをローカル環境にファイルとして保存可能。

文字情報読み取りについては、OCRを使わずに文書から構造化情報を抽出する方針で進めたが、正確な認識が難しかった。

③帳票識別機能

複数種類の財務諸表が束ねられた状態のPDFデータを別々に処理するために、帳票を自動分類する要件。



③帳票識別機能

画像認識ライブラリKERASを利用して、帳票分類を実施。

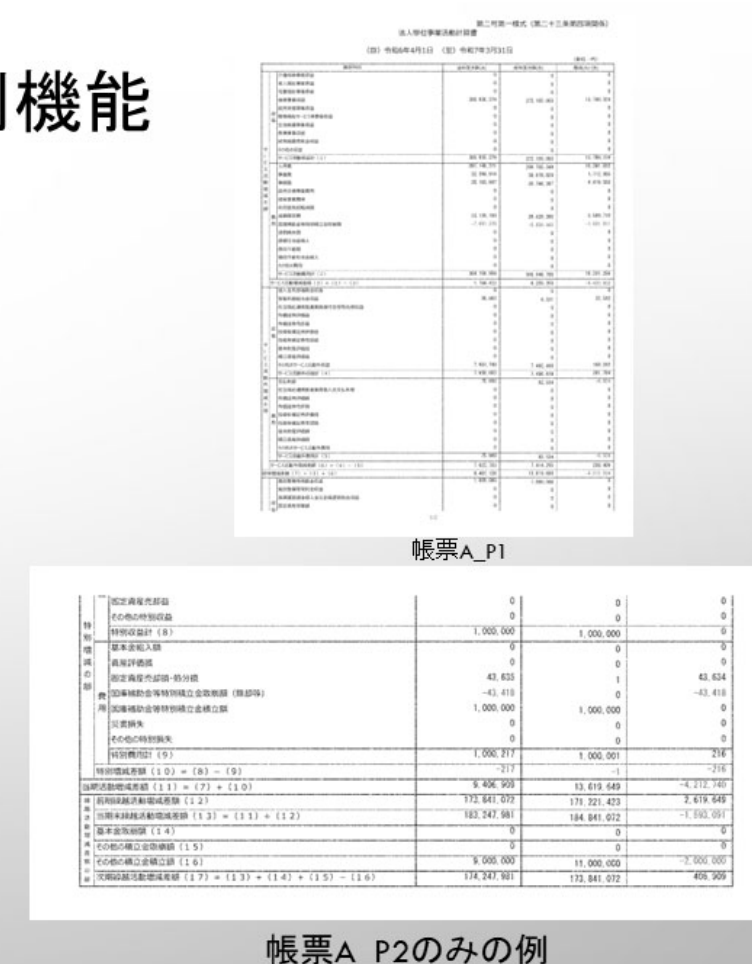
帳票A 56表、帳票B 55表を学習データとして利用。

学習後のモデルを使い学習に用いなかったデータ

(帳票A 26表、帳票B 16表)に対して分類を実施。

結果:
帳票A 26表中16表正しく分類(正解率: 61.5%)
帳票B 16表中16表正しく分類(正解率: 100%)

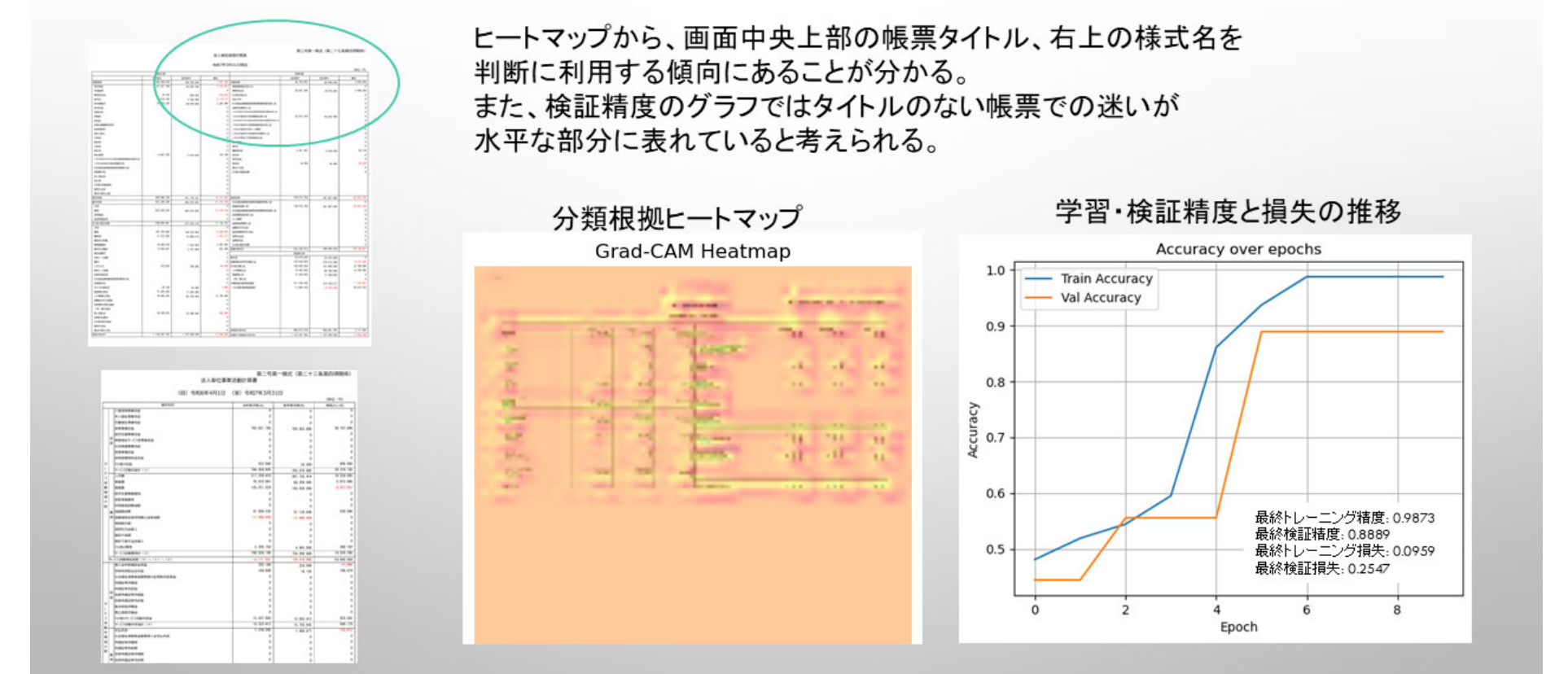
帳票Aのうち誤った分類になった10件のデータを確認したところ、帳票が2ページ目にまたがったため、2ページ目のみのファイルになったものであり、タイトル部分の表示がないために誤分類になったと考えられる。(なお、1ページ目のみの正解率は100%)



帳票A_P2のみの例

③帳票識別機能

ヒートマップから、画面中央上部の帳票タイトル、右上の様式名を判断に利用する傾向にあることが分かる。
また、検証精度のグラフではタイトルのない帳票での迷いが水平な部分に表れていると考えられる。



まとめと課題

- ローカルPCでの動作を実現したことにより、インターネット非接続環境で運用可能。
・セキュリティを確保しつつAIの機能が利用可能
- 学習結果をローカル環境に保存可能。
・帳票の変更や、新帳票の追加時も既存の学習結果に影響を与えず新たな学習が可能。
・年度ごとに学習しなおして履歴管理するなど、柔軟な運用が可能。
- 帳票の識別機能単体の実装が可能。
・事前に帳票分類ができるため、識別機能のない製品も導入候補になり、製品選択の幅が広がる。

これからの課題
・OCRを使わずに文書から構造化情報を抽出するための実装、学習方法の改善。
・donutの初期学習モデルで精度が出なかったためファインチューニングを行う改善。また、LLM自体をローカル環境で動作させる検証。
・インターネット非接続かつコーディングの負担も抑えて画像からの文字抽出の仕組みを構築できる可能性。

制作を通しての気づき

帳票分類や画像からのテキスト読み取りといったAIに関するプログラムの製作自体よりも、学習に利用するデータの整備、またその作業に必要となるプログラム作成も同等あるいはそれ以上の作業負担がかかると感じた。(AI: 70時間、周辺PG: 80時間程度)

【作業に必要となったプログラムの例】

学習用データが大量に必要となるためWEBからデータをダウンロードするためのスクレイピングとZIPからのファイル抽出、PDFファイルの画像化、ファインチューニングのための構造化をJSONファイルの作成(それぞれ約30分/法人分の処理に使用)、OCRによる処理検証用、画像と座標を直してその部分のOCR結果(TESSERACTによる)を返すプログラム。

● 対応として、COPILOTやCHATGPTを積極的に活用した。

copilotやchatGPTによって、プログラムを書くことの効率化が図れる一方、プロンプトに対して再現性のある形で正確な指示を出すことの難しさを体験できた。

【誤った例】

先行PGのファイル出力先フォルダが変わったのに、作成された後続PGは元のフォルダから入力していた。エラー内容の対応が室々通り(データ型エラーの都度、PG間受渡しのデータ型を変えて戻しを繰り返すような回答)。

● 対応として、プロンプトに対してはテンプレート化した書式で指示することで誤り漏れがないようにした。