Center for Economic Institutions

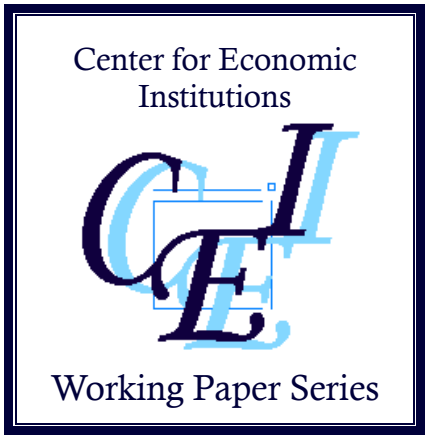
Working Paper Series

No. 2021-3

“Textizing statistical tables using OCR at scale”

Yutaka Arimoto

July, 2021



Institute of Economic Research
Hitotsubashi University
2-1 Naka, Kunitachi, Tokyo, 186-8603 JAPAN
https://cei.ier.hit-u.ac.jp/English/index.html
Tel:+81-42-580-8405/Fax:+81-42-580-8333

OCR を利用した統計表の体系的なテキストデータ化

2021 年 7 月 21 日

有本寛 *

概要

本稿は，OCR を利用して，統計表を体系的かつ大規模にテキストデータ化するための要件と方法を解説する．統計表を OCR でテキストデータ化するには，高い精度の表レイアウト解析が求められる．筆者が開発している ocrstats は，バッチ処理，定型的な工程の自動化，外部 OCR の利用，実用的な精度の表レイアウト解析を実現し，作業効率の改善を図っている．また，ocrstats を使って『日本帝国統計年鑑』をテキストデータ化する過程で得られたノウハウや，パネルデータの作成にあたって変数を経年的にリンクする方法も解説する．

* 一橋大学経済研究所．〒186-8603 東京都国立市中 2-1. Tel: 042-580-8381. Email: arimotoy@ier.hit-u.ac.jp.

謝辞

本稿は，日本学術振興会・二国間交流事業共同研究・セミナー「『行き倒れ』の日韓比較史：『行旅死亡人』データからの接近」（課題番号：JPJSBP120208803，研究代表者：有本寛），および日本学術振興会・人文学・社会科学データインフラストラクチャー構築推進事業（業務主任者：深尾京司・一橋大学特任教授）の成果の一部である．本稿の執筆にあたり，藤岡健太郎，望月祐之，吉野美奈子各氏にシステム構築や校正作業等の支援を頂いた．また，相川雄哉，小谷厚起，銭悠宝，羽生寛明，廣瀬総一郎各氏は非常に優秀なリサーチアシスタントとして，研究を補助して頂いた．記して感謝したい．

Textizing statistical tables using OCR at scale

July 21, 2021

Yutaka Arimoto*

Abstract

This paper describes the requirements and methods for textizing statistical tables using OCR at scale. The major challenge of textizing statistical tables by OCR is analyzing the table layout with high accuracy. I develop a Python tookit, **ocrstats**, that supports the task by providing batch processing, automation of routine processes, use of external OCR, and table layout analysis with practical accuracy. I also explain practical tips learnt from the process of textizing the *Japan Imperial Statistical Yearbook* using **ocrstats**.

* Institute of Economic Research, Hitotsubashi University. Naka 2-1, Kunitachi, Tokyo 186-8603, Japan; Phone: (+81) 42 580 8381; Email: arimotoy@ier.hit-u.ac.jp.

1. はじめに

社会科学分野では，統計的手法を使った実証研究や EBPM（evidence-based policy making）の関心の高まりなどから，統計データ分析の重要性が増している．明治維新以降，日本では政府を中心に膨大な（政府）統計が作成されてきた．こうした統計の利用には常に一定のニーズがあるものの，十分活用されていない．その理由は，統計の多くが紙媒体に印刷された統計表，もしくはよくて電子画像で提供されており，統計データとしてコンピュータで利用できる情報は，比較的近年のものに限られるからである（以下，紙媒体の統計情報を「紙統計表」，画像としてデジタル化された統計表を「統計表画像」，機械判読可能な統計データを「テキストデータ」と呼ぶ [1]）．統計データとしてコンピュータ利用するためには紙統計表の電子画像化，およびテキストデータ化が必要だが，それには膨大な時間とコストがかかる（山崎 2018；五十嵐 2020）．また，これを体系的にテキストデータ化するノウハウの蓄積や共有が進んでいないことも，参入障壁となっていると考えられる．

本稿は，統計表を体系的かつ大規模にテキストデータ化する作業の要件と課題を整理し，これを支援するために筆者が開発中のツール（ocrstats）の概要を解説する．ocrstats は，統計表画像の体系的なテキストデータ化を支援する Python 言語のパッケージである．バッチ処理，定型的な工程の自動化，OCR の利用，独自の表レイアウト解析アルゴリズムの実装を通して，作業効率の改善を図っている．特に，既存の OCR で課題であった表復元（統計表の表レイアウト構造を正確に解析し，認識した文字列を適正なセルに配置すること）を，実用的な精度で実現できた点がブレークスルーである．現在，ocrstats を使って，『日本帝国統計年鑑（現『日本統計年鑑』）』（1905-40 年分）などの統計を体系的にテキストデータ化し，府県レベルのパネルデータを構築する作業を進めている．この作業過程で得られたノウハウを共有する．これによって，大規模に（公的）統計をテキストデータ化する道筋を示し，その推進に寄与することが狙いである．

構成と概要は以下の通りである．第 2 節では，OCR を使って統計表画像をテキストデータ化する際の課題を解説する．第 3 節で，統計表画像をテキストデータ化する作業の流れを概観する．第 4 節では，ocrstats がどのような支援機能を提供するか，テキストデータ化の作業工程に沿って紹介する．また，表レイアウト解析のアルゴリズムも解説する．第 5 節では，『日本帝国統計年鑑』のテキストデータ化作業を紹介する．さらに，パネルデータの作

[1] テキストデータ化を「電子化」と呼んでもよい．実際，「電子化」や “digitize”という場合，テキストデータ化を意味していることが多い．ただ，電子画像とすることを「電子化」と呼ぶこともある．近年，各種資料のマイクロフィルムの電子画像化が進み，JAPAN SEARCH や J-DAC などのデジタルアーカイブや，国会図書館デジタルコレクションで提供されていることが多い．これを受けて，「○○は電子化されている」と称されることがあるが，実際は電子画像化に留まっている．本稿ではこうした混同を避けるため，電子画像化とテキストデータ化を区別し，両者を合わせて電子化と呼ぶことにする．

成にあたって，変数を経年的にリンク（接続）し時系列データ化する「系列化」作業の要件と方法を解説する．最後に第 6 節で，課題と今後の展望をまとめる．

2. OCR の活用と課題

OCR（optical character recognition：光学文字認識）は，画像から文字を認識し，文字コードに変換する技術である．OCR を利用することで，データ入力の手間が省け，作業効率を大幅に高められる可能性がある．ただし，統計表は文字（数値）情報だけでなく，表のレイアウトが重要な意味を持つ．文字列を正確に認識すると同時に，その文字列が表のどのセルに位置するものなのかを特定し，適切に配置することが求められる．このため，文字認識だけでなく，表のレイアウト構造を正確に解析し，改行・改列位置・セルを特定する表レイアウト解析（後述）が必要となる．

近年，深層学習などの AI 技術が OCR にも応用されている．その結果，市販の OCR パッケージソフトやクラウドサービスの文字認識や表レイアウト解析の精度も向上しているとみられる．表レイアウト解析は各サービスとも，特にビジネス用途で OCR のニーズの高い領収書や請求書，身分証明書などの帳票について，高い解析精度を謳っている．ただ，既存の表レイアウト解析をただちに統計表に利用するのは難しい．既存サービスは，レイアウトが固定された帳票（定型帳票）であったり，あらかじめ画像のどの位置にどの情報が記載されているかを指定するテンプレートを作成する必要があったり，フォーマットが異なる帳票（非定型帳票）ではあるものの，ある程度共通性・規則性のある帳票（請求書など）を機械学習させてテンプレートを作成したものであったりと，利用できる帳票が限定されているのが現状である．統計表はこれらの帳票と比べて，表ごとにレイアウトが異なっていたり，構造が複雑だったり，複数ページに渡って表が続いたりするといった固有の難しさがある（Raja, Mondal, and Jawahar 2020, 2）．

OCR を使った統計表画像のテキストデータ化には，文字認識と表レイアウト解析の順序によって，2 つのアプローチがある [2]．ひとつは，表レイアウト解析→文字認識をおこなうトップダウン型のアプローチである．表レイアウト解析し，特定されたセルごとに順次画像を切り出して OCR 処理し，認識された文字列を元のセル位置に配置して，表復元する．いまひとつは，文字認識→表レイアウト復元をおこなうボトムアップ型アプローチである．表レイアウト構造をあえて無視したうえで，画像の文字列を網羅的に走査し，認識された文字列とその位置情報（文字列を囲む枠線（バウンディングボックス）の座標）を取得する．そのうえで，位置情報から表レイアウトを解析し，表復元する．

[2] くずし字 OCR におけるトップダウン・ボトムアップ型のアプローチの相違については，北本・タリン・Lamb・Bober-Irizar (2019, 230 頁)を参照.

トップダウンのアプローチは，最終的に出力される表が，前工程の表レイアウト解析の精度に強く依存する．表レイアウト解析の精度が低いと，空行や空列が混じったり，行や列の途中でセルがずれたりして，修正するのに大きな手間がかかる．ボトムアップのアプローチは，文字列の範囲特定の精度に強く依存する．いかに正確に，表内の各セルの文字列を独立した範囲として認識するか（上下や左右のセルの文字列を連結して，ひとかたまりの文字列として認識しないか）がポイントである．この精度が十分高ければ，各文字列の位置情報から統計表の改行・改列の位置座標を特定し，表レイアウトを再構成できる（後述）．さらに，その再構成も何度でも柔軟に修正できるため，事後的に精度を高めることができる．

3. 統計表のテキストデータ化作業の流れ

本節では，統計表をテキストデータ化する作業の流れを概観する（**図 1**）．

1. 統計表画像の作成. 紙統計表を撮影ないしスキャンし，デジタル化する．このステップの精度は後工程に響くため，(a) スキャンの解像度は 200dpi 以上とする，(b) できるだけ水平にスキャンする，(c) 紙統計表の中心と画像の中心を合わせる，ことが望ましい．

2. 画像補正. 後工程の精度を上げるため，水平と歪みの補正をおこなう．統計表画像が見開きでスキャンされている場合，左右の頁で「＼／」ないし「／＼」のように，逆方向に傾いていることが多い．このため，見開き画像をのど元で分割し，片ページずつ水平補正をおこなう．資料画像の可読性を高めるため，白色化処理をしてもよい [3].

3. アノテーション. 画像のどこが統計表で，どこが統計表の表題や表頭，表側，表体，注釈かを解析（レイアウト解析）し，範囲指定したうえでラベルづけする（アノテーション）．

4. OCR 処理する範囲の画像の切り出し（クロップ）.（見開きの）資料画像には複数の統計表が映っていることも多いため，統計表ごとに切り出す．あるいは，統計表全体ではなく，表体のみを切り出してもよい（後述）．

5. OCR 処理. OCR エンジンで画像を解析し，文字（列）を認識し，その位置情報を抽出する．

6. 表レイアウト解析. 表のレイアウト構造を解析し，行・列・セルを特定する．

7. 出力. 認識された文字列を，表レイアウト解析で特定した適正なセルに配置し，スプレッドシート上に表として出力する．また，検証用の画像も出力する．

8. 校正. 認識された文字列が原本通りか，セル配置が正しいかを検証し，修正する．

[3] 白色化の有無が OCR 精度を左右するかは，今後の検証が必要である．体感では大きな影響はない．

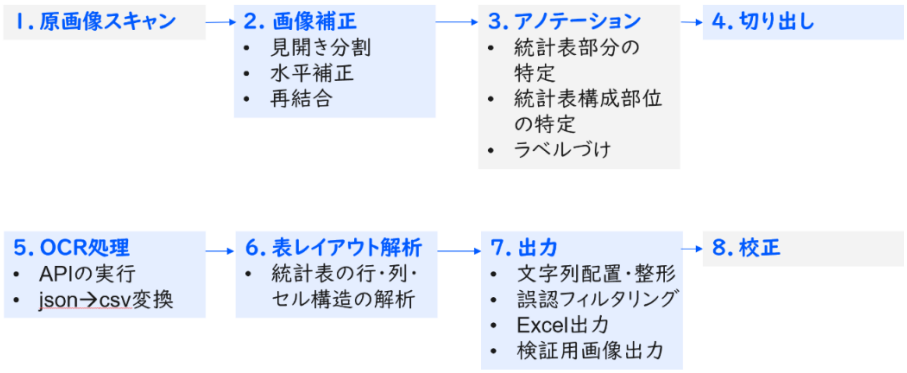


図 1. 統計表のテキストデータ化作業の流れ

注）青塗りされた工程は ocrstats で自動処理できる.

4. ocrstats の概要

本節では，筆者が開発中のツール(ocrstats)を解説する．ocrstats は，統計表画像のテキストデータ化を体系的に支援する Python 言語のツールである．紙媒体もしくは電子画像の状態の統計は，政府統計だけでも無数にある．数百から数千単位の統計表を体系的かつ大規模にテキストデータ化することを見据え，開発にあたって，(a) 一括処理（バッチ処理）ができること，(b) 定型的な工程をできる限り自動化すること，(c) OCR を活用すること，を要件とした．実用的な精度で（とりあえず）動いて使えるものを構築すること重視し，実装にあたって使える外部ツールはできる限り使う．

ocrstats は Python パッケージとして作成している．NumPy や Pandas，SciPy といった一般的な拡張ライブラリのほか，画像処理ライブラリとして OpenCV2 を使っている．また，アノテーションツール（VoTT）と OCR エンジン（CLOVA OCR）を外部ツールとして組み合わせて利用する [4]．開発は，Windows 10 Pro 上で，拡張ライブラリも含めて miniconda で容易に構築できる Python 環境でおこなっている．以下，第 3 節で概観した作業の流れにしたがって，ocrstats がどのように作業を支援できるかを説明する．

4.1. 画像補正

ocrstats は画像補正のため，(a) 画像を見開き分割し保存，(b) 画像を水平補正，(c) 分割・水平補正した画像を再結合する関数を，それぞれ提供している．見開き分割は，資料ののど元（中心）が画像上どこにあるかを判別する必要がある．現在は，分割位置を指定する

[4] 後述するように，これらの外部ツールは VoTT や CLOVA OCR に限定する必要はない.

引数を与えられるようにしている[5]．水平補正は，投影プロファイルに基づいて歪み角度を特定するアルゴリズムを使っている[6]．傾き角度を探索する範囲と，探索角度の精度の 2 つの引数を与えて傾き角度を特定し，補正する．反復ループ処理となるため，与える引数にもよるが，処理速度は遅い．

4.2. 画像のアノテーションと切り出し

次のステップは，OCR 処理する画像の範囲（統計表部分）の切り出しである．それには，画像から統計表部分や，統計表部分のなかから表題，表頭，表側，表体，注記などの各構成部位の領域を識別する，レイアウト解析が必要である．現時点では，ocrstats は自動的なレイアウト解析を実装していない．人の目で統計表を特定し，アノテーションアプリケーションを使ってその範囲指定，および表頭，表側，表体，注などラベル付けを手動でおこなう（アノテーション）[7]．このレイアウト解析とアノテーションは，大きなボトルネックではない．アノテーションする箇所が少なく（ひとつの統計表につき，多くは 1 つの表体しかない）[8]，ツールが大量の画像の効率的なアノテーションに必要な機能を備えており，統計表ひとつにつき 1 分もかからない．

アノテーションアプリケーションは，アノテーションした領域のラベルと座標情報を出力できる．ocrstats はこの情報をもとに，画像を自動で切り出す[9]．

4.3. OCR 処理

OCR 処理は，外部の OCR エンジンを利用する．OCR エンジンは，(a) 認識した文字列とそのバウンディングボックスの位置情報が得られる，(b) API やコマンドラインインターフェース（CLI）による操作を受け付けて，バッチ処理との連携がとれる，(c) 文字列範囲の認識精度が高い，という要件を満たせば基本的には何でもよい[10]．

---

[5] デフォルトでは画像幅の中心（半分）で左右に分割する．引数で，左右それぞれ，分割する位置を画像幅の比率で指定できる．例えば，左から 0.52，右から 0.48 であれば，画像の中心からやや右に寄ったところで分割する．余裕をもってどちらも 0.52 程度に設定しておけば，中心が多少ずれていても，ページ途中で切り出されることはない．

[6] https://stackoverflow.com/questions/57964634/python-opencv-skew-correction-for-ocr

[7] アノテーションツールは，画像上の範囲指定と，指定範囲の座標情報が得られれば，何でもよい．フリーのツールとしては，ほかに LabelImg などがある．

[8] この単純さが，多数のタグづけと複雑なレイアウト解析とが必要な，『人事興信録』や企業会計報告書集（Shen, Zhang, and Dell, 2020）との違いである．

[9] 切り出しは，OS に付属の画像加工ソフトウェアなどを使って手動でおこなってもよいが，アノテーションツールを使う方が利便がよいだろう．切り出し範囲の修正などの手戻りが容易であり，画像ファイルが増えて記憶媒体の容量を圧迫することもない．

[10] クラウド型汎用 OCR エンジンとして CLOVA OCR，Google Cloud Vision，Amazon Textract，市販の OCR パッケージソフトとしては「読取革命 Ver.15」，「e.Typist v.15.0」，「ABBYY FineReader PDF」などがある．

ocrstats では現時点では CLOVA OCR を利用しており，切り出した画像を自動的に API に送信・実行する関数を提供している．なお，CLOVA OCR では，処理結果が json 形式で返ってくる．このままでは可読性が低いため，整形して csv 形式に変換する関数も備えた [11].

4.4. 表レイアウト解析

表レイアウト解析では，ボトムアップ型のアプローチを採用し，統計表の改行・改列の位置座標を自動検出するアルゴリズムを実装した．手順は次の通りである（**図 2**）.

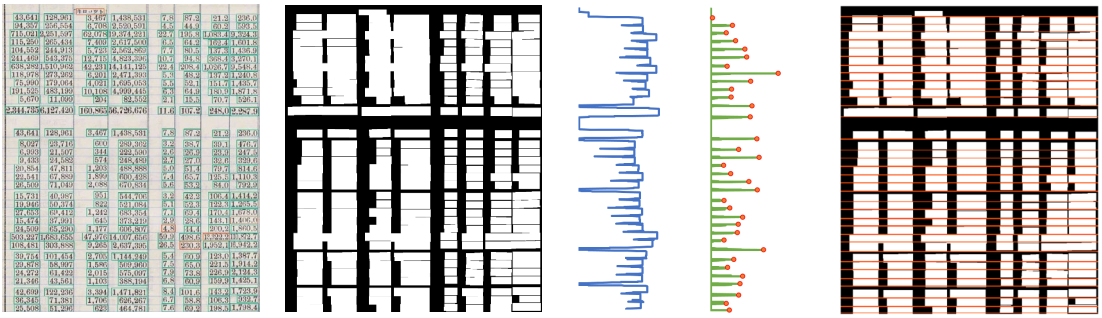
- (a) 統計表画像を OCR 処理し，文字列とバウンディングボックスの座標情報を得る．
- (b) バウンディングボックスの座標情報を使い，各バウンディングボックスを塗りつぶしたモノクロ画像を作る．
- (b) 水平・垂直投影プロファイルをとる．水平・垂直方向に，塗りつぶしたバウンディングボックスのピクセル数を合計し，ヒストグラムを作成する作業である．
- (c) 次に，ヒストグラムの一つ前の座標との差分（変化）を計算する．水平方向（行方向）でみたとき，文字列（バウンディングボックス）が多い y 座標では，ヒストグラムが正で大きな値をとるが, 改行すると文字列がないのでヒストグラムはゼロ（に近い値）をとる．しがたい，差分をとると急減する．急減のみに注目したいため，正の変化はゼロに置換し，さらに変化を正で定義するため絶対値をとる．
- (d) このヒストグラムの変化（急減）のピークを検出し, 改行位置の座標を特定する [12]. ピーク検出は，ピークの高さと幅の 2 つのパラメータで調整している．
- (e) 検出したピークに基づき，水平罫線を描画する．

同様の作業を列方向におこなうことで，改行・改列位置が特定され，表レイアウトが再現できる．

市販パッケージは導入費用が安いが，必要要件を満たせず，現時点では利用は除外している．ほかに，オープンソースの OCR エンジンである Tesseract を使う選択肢もある．表復元の工程の精度が要件（c）に強く依存しており，その精度が高いという観点から，ocrstats では CLOVA OCR を利用している．

[11] なお，以降の工程は，認識されたテキスト，信頼度，バウンディングボックスの座標情報を含む csv ファイルがあればよい．これらの情報を含む csv ファイルが作成できれば，OCR エンジンは何でもよい．

[12] 画像が完全に水平で，ノイズがまったくなければ，ヒストグラムが 0 である位置（の例えば中心）を改行位置と推定できる．しかし，活字ベースの統計表のスキャン画像は歪みがあり，垂直罫線があることも多いため，この手法は使わなかった．



(a) (b) (c) (d) (e)

図 2. 表レイアウト解析の手順

注）(a) OCR 処理し，認識した文字列のバウンディングボックスを描画．(b) バウンディングボックスを塗りつぶしたモノクロ画像を作成．(c) 水平投影プロファイルの作成．(d) 水平投影プロファイルの差分をとり，ピーク検出．(e) 検出したピークを改行位置（座標）とみなし，水平罫線を引く．

ポイントは 3 つある．第 1 は，原画像ではなく，バウンディングボックスのモノクロ画像で投影プロファイルをとることである．バウンディングボックスを使うことで，投影プロファイルのヒストグラムが断続的に変化する．このため，差分をとったときの変化が際立ち，ピーク検出がしやすくなる．このようなボトムアップ型のアプローチが使えるのは，OCR エンジンが文字列範囲を高精度で特定できるからである．

第 2 に，大きさが異常なバウンディングボックスを除外する．表体の文字列は，おおむね同じような大きさの横長のバウンディングボックスとなる．しかし，表頭は変数の説明であり，幅や高さが大きくなりやすい．表側も都道府県名などを均等割り付けしてあることが多く，行方向で文字間隔が開くのに対して，列方向では詰まっているため，OCR は列方向の縦書きと誤認識し，縦長のバウンディングボックスができてしまう（後述）．

第 3 は，ピーク検出のパラメータ調整である．改行と改行の間は，平均的な文字列の高さ（バウンディングボックスの平均的な高さ）以上離れているだろう．また，ピークの高さ（ヒストグラムの急減幅）も，バウンディングボックスの平均的な幅程度はあるだろう．実際に改行は，ピーク検出する幅のパラメータを，バウンディングボックスの高さの中央値以上に設定することで，おおむね良好な結果を得ている．

4.5. 出力

最後に，認識した文字列を適正なセルに配置し，表形式で出力する．文字列配置は，文字列の重心が含まれるセルに配置した．具体的な手順は次の通りである．

- (a) 各文字列のバウンディングボックスの重心を求める．

- (b) 各バウンディングボックスに，配置する行と列の番号を付与する．重心座標に対して，表レイアウト解析で特定した行と列の座標を境界値としてビニング処理することで，容易に付与できる．
- (c) ひとつのセルに，複数の文字列が配置される場合は，当該バウンディングボックスたちを左上から右下への順に並び替え，文字列を連結した．
- (d) 最後に，各バウンディングボックスに付与されたセルの位置をもとに，文字列を並び替えて，Excel 形式で出力した．

なお，別途用意した csv 形式の正誤対照表を読み込み，一括置換するフィルタリング関数も用意した．これによってありがちな OCR の誤認識を機械的に修正できる [13]．

また，ocrstats は検証用画像を出力できる．認識精度が低い，文字列が含まれる，バウンディングボックスの大きさや形状が異常である，といった確認を要する文字列について，色を変えたバウンディングボックスを資料画像にオーバーレイ描画したものである．これによって，OCR の認識漏れや要確認箇所を目視で特定しやすくなる．

4.6. 校正

校正作業は基本的に人の目に頼らざるを得ない．セル配置の確認は，表復元の工程の精度が高ければ，微修正ですむ．一方，誤認識の確認と修正はテキストデータ化でもっとも手間と時間がかかる工程である．

もっとも正確に校正するには，出力されたデータを読み上げて [14]，原資料と突合する必要がある．ただし，この方法は時間と人手とコストがかかる．別の簡易な方法は，サムチェックによる校正である．統計表には集計欄があることが多い．OCR 処理し整形した出力データを行・列ごとに合計し，それと集計欄の数値が合致しているか確認すればよい．合致していない場合は，その行または列のどこかに誤認識があるか，原本の数値ないし集計欄そのものに誤りがある．行と列それぞれの方向に集計欄があれば，双方で合致していないセルが誤認識の箇所であり，特定が容易である．この方法では誤認識を完全に検出することはできないが，残る誤認識はランダムな誤差の範囲とみなすこともできよう．

主だったエラーは，文字列の認識範囲の不良と，認識漏れである（**図 3**）．認識範囲不良は，原本画像に表レイアウト解析で特定した改行・改列位置座標で水平・垂直罫線を引いた画像を作成し，それを OCR 処理することで改善できた（**図 4**）[15]．

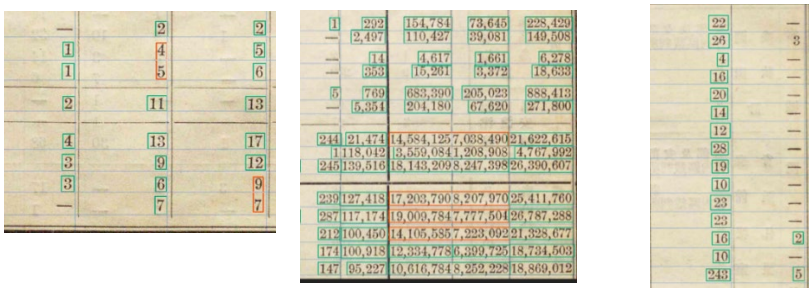
---

[13] 例えば，「エ」と「1」，「g」と「9」，「G」と「6」，「S」と「8」，「ク」と「7」，「ち」と「5」などがよく誤認識される．これらの正誤対照表を csv ファイルにまとめ，フィルタリング関数で一括置換する．

[14] OS の読み上げ機能を使えば，追加的な人手はかからない．読み上げ速度の調整もできる．

[15] CLOVA OCR では，罫線にある程度（3 ピクセル）の太さがないと，文字列を分割して認識しない．ただし，罫線を太くすると文字列と重なり，認識精度が悪化することがある．この場合は，行や列，場合によってはセルで画像を切り出しして OCR 処理すればよい．

文字列そのものの認識精度は，特に表体の数値部分についてはほぼミスがない．誤認識も典型的なものはフィルタリング関数で置換すればよい．表頭や表側の漢字・かな・カナは，旧字体が混在する場合，修正が必要となることがある．



(a) (b) (c)

図 3. 主なエラーの例

注）(a) 認識範囲不良（「縦読み」）．(b) 認識範囲不良（2 つの文字列を横に連結して認識）．(c) 認識漏れ（2 列目の上から 2 行目の「3」が認識されていない）．

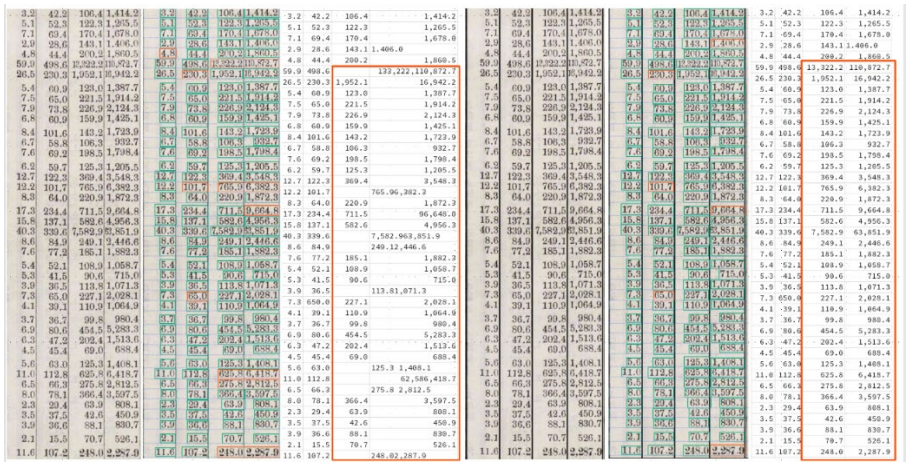


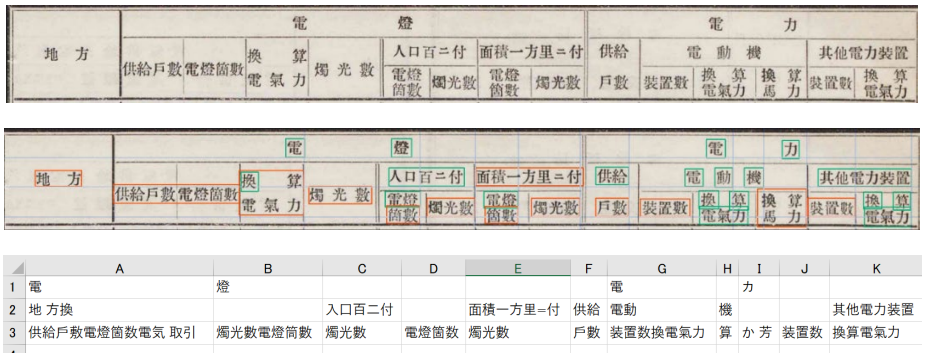
図 4. 文字列範囲認識不良の回避

注）左半分は原本をそのまま処理した．一部で文字列範囲認識不良を起こし，2 つの文字列が連結されている．右半分は，原本画像に垂直罫線を引き，OCR 処理したもの．範囲認識不良が起きず，適切に認識されている．なお，原本にはもともと垂直罫線が入っているが，やや薄い．

4.7. 表頭と表側の処理

表体はほぼ問題なくテキストデータ化できるが，表頭と表側は OCR の文字列範囲の認識が難しく，調整が必要である．表頭は，入れ子構造が複雑であり，実用レベルの精度でのテ

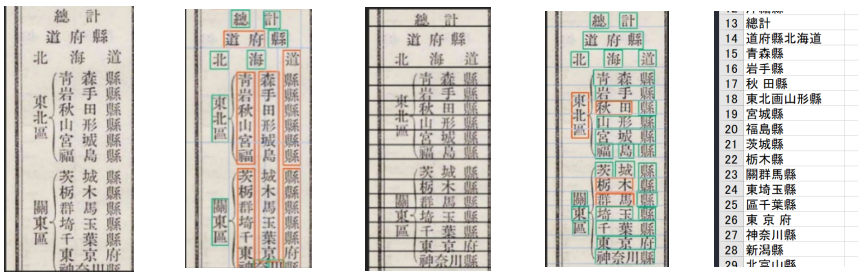
キストデータ化はできていない（**図 5**）.



	A	B	C	D	E	F	G	H	I	J	K
1	電	燈					電		力		
2	地 方換			入口百二付		面積一方里=付	供給	電動	機			其他電力装置
3	供給戸敷電燈箇数電気 取引		燭光数電燈筒数	燭光数	電燈箇数	燭光数	戸數	装置数換電氣力	算	か 芳	装置数	換算電氣力

図 5. 表頭のテキストデータ化の例

同様の問題は表側でもみられるが，こちらはおおむね解決できた．表側は，前述の通り，横書きを縦書きと誤認識（「縦読み」）してしまう（図 6(b)）．そこで，表体の表レイアウト解析で特定した改行位置座標で水平罫線を引いた画像を作成し（図 6(c)），これを OCR 処理することで，ある程度横書きとして認識させることができた（図 6(d)）．校正が必要だが，実用範囲内だろう（図 6(e)）.



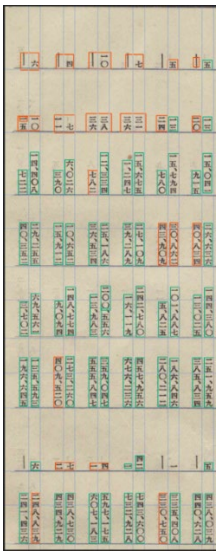
(a) (b) (c) (d) (e)

図 6. 表側の OCR 処理

注）(a) 原画像．(b) 原画像を OCR 処理したもの．「縦読み」している．(c) 水平罫線を引く．(d) 水平罫線を引いた画像を OCR 処理したもの．(e) 整形済み Excel 出力.

4.8. 縦書き・漢数字の統計表の処理

縦書き・漢数字の統計表は，文字認識精度にやや難があるものの，表レイアウト解析と文字列配置については，ほぼ実用レベルの精度で整形できている（**図 7**）．漢数字はローマ数字の変換フィルターを介せばよい.



	六		四		18		10		五	:
一	する		二 七		美味	素三		1/3	一三	二〇
七二二	一四、四〇八	三九〇	六、〇二六	七八二	一二、三三四	一、二四七	一五、六七五	七八〇	一五、七九四	九一五
四〇、三五二	二九、二五五	一五、九一二	一〇、六五二	三六、五三四	二五、一八六 二〇〇	三九、二八九	二七、一〇九	四三、九〇九	三〇、八六二	四〇、八三四
三、七〇二	六九、五六一	九、〇九四	一四八、七七四	一三、九八三	、五五六	一六、一一九	二四二、七八〇	五、八二五	一〇一、八八七	一三、〇二五
一九六、六四五	一三五、五九三	四〇九、五二〇	二七三、二六〇	五五五、八四七	三五九、〇四七	六七六、二三六	四五七、九五六	二八〇、二一二	一八六、八四六	三八五、八三四
	六		二 七		一四	1 四				
二四一、四三六	二四八、八三九	四三四、九二九	四三八、七三〇	六〇七、一八三	五九七、一七五	七三二、九二八	七四三、六〇〇	三三〇、七五〇	三三五、四〇八	四四〇、六二八

図 7. 縦書き・漢数字の統計表のテキストデータ化の例

5.『日本帝国統計年鑑』のテキストデータ化作業の概要

現在，ocrstats を使って，『日本帝国統計年鑑』（1905-40 年，36 冊），『人口動態統計』，『都道府県別農業基礎統計』の一部統計のテキストデータ化が進行中である．本節では，特に『日本帝国統計年鑑』の作業過程を解説する．

5.1. 資料の概要

『日本帝国統計年鑑』（1882～）は，各省庁が収集した主要公的統計をまとめた，日本の代表的かつもっとも基本的な統計書である．現在も『日本統計年鑑』として継続的に刊行されている．収録されている分野は，国土・気象，人口，農林水産業，鉱工業，運輸，情報通信，貿易，商業，金融保険，教育，保健衛生，社会保障，財政，公務員，選挙，司法，警察，軍事，国際統計など多岐に渡る．また，統計によっては府県別の情報が得られる．

規模を概観すると，1905-40 年までの 36 年分で約 22,000 表（1 年平均 610 表）ある．現在，進めている作業では，一部分野を対象から外し[16]，さらに都道府県別の情報が得られる統計表のテキストデータ化に限定している．これにより，36 年分全体で約 4,200 表が対象となる．

テキストデータ化に使っている電子画像は，専門業者が原資料の冊子を解体しスキャンした高精度のものである．歪みも少なく，原本と画像の中心もほぼ一致している．プロジェクトでは，1 年（1 冊）ごとに PDF ファイルになった画像データを使用している．

[16] 人口，農林水産業，貿易，植民地，国際統計は除いている．人口，農林水産業は，別に利用可能な統計がある（『人口動態長期統計時系列データ』，『都道府県別農業基礎統計』）．

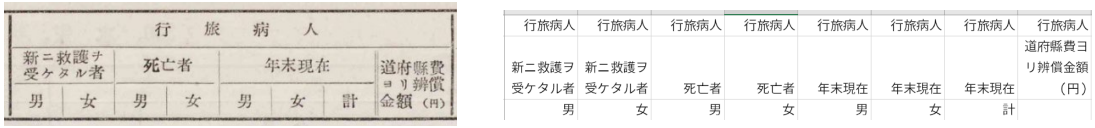
5.2. 作業過程での留意事項

作業にあたっての一般的な留意事項を列挙する．

- 水平画像補正は，傾き角度を探索する範囲と精度にもよるが，ループ処理のため時間がかかる．プロジェクトでは，探索範囲を 2 度，探索精度を 0.02 度に設定した．この設定で画像 1 枚当たり約 2 分かかっている．このため夜間に処理をおこなっている．なるべく水平にスキャンしておけば，探索範囲を狭められるため，処理時間を短縮できる．
- アノテーションは，オープンソースの画像アノテーションアプリケーションである VoTT を利用している．OCR 処理する画像の範囲は，表体のみとした．OCR エンジンによる表頭や表側の文字列範囲の特定精度が悪く，それが表レイアウト解析の際にノイズとなるからである．
- 表頭と表側は OCR 処理から除外しているため，手入力となる．表頭は入れ子構造になっており複雑なため，やむを得ないと考えている．表側は原則として，都道府県名であり，コピー＆ペーストで対応できる．ただし，年や統計表によって，都道府県の並び順が違っていたり，地区ごとの小計行が挟まっていたり，該当しない場合は行そのものが省略されていたりするため，校正時の確認は必要である．なお，4.7 節の方法で実用的な精度で自動テキストデータ化できるが，表側部分のアノテーションが必要である．
- 校正を終えた統計表のテキストデータは Excel 形式で出力している．統計表のレイアウトは次の原則に則って指定した．第 1 は，機械判読可能性を担保することであり，総務省統計局（2020）「統計表における機械判読可能なデータ作成に関する表記方法」のガイドラインにしたがっている．表頭が複数行に分かれることを許容するかは意見の割れるところだが，表頭の複雑な入れ子構造を把握するため，現時点では許容している（**図 8**）．校正の過程でサムチェックの行や列などが追加されるが，これは適切な行インデックス[17]を付与し，統計処理する際は機械的に除外できるようにした．第 2 は，できる限り原本の統計表を忠実に再現することである．原本との参照可能性を担保するためである．このため，原本上の明らかな誤植もそのまま残す．したがい，修正する場合は，利用者が各自で統計処理スクリプトを使って明示的におこなう．
- データのシートのほかに，メタデータのシートを作成し，そこに表体の開始行，掲載年，統計表番号，表題，表注，注釈（サムチェックの不整合など，校正時に気づいた点など），更新履歴を記録することとした．
- 表体の開始行を記載しておくことで，表頭（変数名）とデータの位置を機械的に読み取ることができる．これにより，表頭の処理（複数行に分かれた入れ子構造を連結する）

[17] 行は都道府県でインデックスされる．サムチェックの行は，都道府県を指定する変数（列）に都道府県名の代わりに「check」といった特定の文字列を入れた．統計分析の際は，都道府県名が「check」の行を除外すればよい．

や，データのみの読み取りといった作業が容易になる[18].



行旅病人							
新ニ救護ヲ受ケタル者		死亡者		年末現在			道府縣費ヨリ辨償金額（円）
男	女	男	女	男	女	計	

行旅病人	行旅病人	行旅病人	行旅病人	行旅病人	行旅病人	行旅病人	行旅病人
新ニ救護ヲ受ケタル者	新ニ救護ヲ受ケタル者	死亡者	死亡者	年末現在	年末現在	年末現在	道府縣費ヨリ辨償金額（円）
男	女	男	女	男	女	計	

図 8. 入れ子構造の表頭の例

5.3. 系列化

『日本帝国統計年鑑』に限らず，公的統計は一般に同じ情報（変数）が経年的に得られることが多い．以下，同じ変数を経年的にリンク（紐付け，接続）し，時系列データにすることを「系列化」と呼ぶことにしよう．研究用途では，都道府県別の横断面データと時系列データを合わせた，府県×年パネルデータとしての需要が高い．このため，プロジェクトでは 1 冊ずつ年ごとにテキストデータ化するのではなく，優先順位の高い変数ごとに（年度を超えて）作業を進めることとした．

変数単位で作業をするためには，目的の変数が各年のどの表に掲載されているかを特定する必要がある．変数 A の表記が経年的に一貫しており，かつ必ず表題 N の統計表に記載されていれば話は単純で，各年の表題 N の統計表をテキストデータ化し，変数 A をリンクすればよい．しかし実際には，年によって（a）変数の定義が変わる，（b）変数名（表頭）の表記が揺れる（A が A’になる），（c）表題の表記も揺れる（表題 N が N’になる），（d）表が分割したり，統合されたりする，といった問題がある．特に（d）は，表単位ですら表題を手がかりにリンクできないことを意味する．変数 A が表題 N の統計表にあると思っていたら，ある年から表題 N の統計表が，表題 N と N’（表題の表記が揺れると N’や N”など）のふたつの統計表に分離し，どちらにあるか分からないという状況が生じる．

このため，関連する複数の表を含む広めの「系列」単位で，経年的にリンクすることにした．先の例だと，表題 N，N’，N”の統計表をひとつの「系列」とする．そうすれば，目的の変数 A は，系列内の統計表のどこかにはある．

系列を作成する具体的な手順は次の通りである．

- (a) 全統計表の目録を作成する．1 行につきひとつの統計表について，掲載年，分野，表番号，表題，統計書上および PDF ファイル上の開始ページなどの情報をリストにしたものである．以下，この統計表目録の各行（統計表）に「系列名」の情報を付与する．
- (b) 1915 年を基準年とし，「系列」名を定め，その系列に属する統計表を特定する．例え

[18] 例えば，データ開始行が 3 であれば，1〜2 行目は表頭である．同じ列の 1 行目と 2 行目の文字列を連結して，入れ子構造をフラットにした変数名を作成したり，3 行目以降の範囲を読み込むことで，データのみにアクセスできる．

ば「学齢児童」系列には，「学齢児童」，「不就学学齢児童」，「小学校及学級」といった表題の表が含まれる（表題は年によって異なる）．

- (c) 年ごとに，基準年の各系列に含まれる統計表と同じ情報が含まれる統計表を拾い出す（これを「拾い出し」と呼んでいる）．統計表目録に系列名の列を追加し，拾い出した統計表の行の「系列名」列に，(b)の系列名（上記例では「学齢児童」）を入力する．

以上の作業により，統計表目録を系列名で絞り込むと，その系列に含まれる統計表の一覧が得られる．以降，系列単位での作業が可能となる．まず，系列ごとに，PDF ファイルから該当する統計表が含まれるページを画像抽出した．統計表目録に，PDF 上の開始ページの情報を付加しておいたため，この抽出作業は自動化している．次に，水平補正，アノテーション，画像切り取り，OCR 処理，校正と進む．校正作業が終わった段階で，t 年の表題 N として独立した統計表群が完成する．

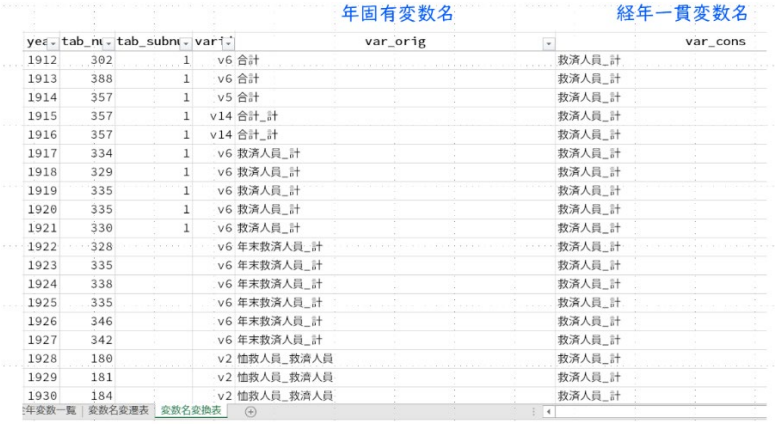
続いて，系列化をおこなう．どの変数を同一とみなし，経年的にリンクするかは，統計の定義や研究者の解釈によって異なる見解がありえる．このため，(a) どの変数をリンクしたかを確認できる透明性，(b) 異なるリンクに容易に切り替えられる柔軟性，を確保することを要件とした．系列化の手順は次の通りである（**図 9**）：

- (a) 「全年変数名一覧」を作成する．これは「系列」に含まれるすべての年のすべて統計表の変数名を一覧にしたものである．1 行につき掲載年 1 年分のすべての変数名を表示する．
- (b) 「変数名変遷表」を作成する．これは全年変数名一覧の情報を組み替えて，同一内容の変数の変数名が，経年的にどう変遷した（と判断した）かを把握するための表である．まず，一覧表の 1 行目をタイトル行，つまり経年的に一貫した統一の変数名（以下，経年一貫変数名）の行とし，適切な経年一貫変数名 A を決める．次に，A と同一内容の変数を各年特定し，同じ列に配置する．t年では A’，$t+1$年では A”とだったとすると，A と同じ列に A’，A”を配置するよう組み替える．
- (c) 「変数名対照表」を作成する．これは，年ごとに固有の年固有変数名と経年一貫変数名を対照させた表である．上記の例では，t年では A’=A，$t+1$年では A”=A，といった対照関係を表す．
- (d) 変数名対照表をもとに，年ごとに年固有変数名を経年一貫変数名に置換する．t年では A’を A に，$t+1$年では A”を A に置換する．これによって，すべての年について，共通の経年一貫変数名がついた統計表が作成される．
- (e) 経年一貫変数名に統一された年ごとのファイルを縦に連結（append）する．

以上で，パネルデータが作成できる．前述の要件（a）（b）は，変数名変遷表を確認または改訂することで実現できる．変数名変遷表の作成・確認・改訂は，どの変数をリンクする

かを目視しながら判断する手作業となるため，Excel ファイル上でおこなう．それ以外の工程は処理スクリプトを用意した．全年変数一覧表を作成する Excel VBA スクリプト（工程(a)），変数名変遷表から変数名対照表を作成する Excel VBA スクリプト（工程(c)），変数名対照表に基づき各年の統計表の変数名を経年一貫変数名に置換，全年分の統計表を連結，パネルデータとして出力する R 言語スクリプト（工程(d)(e)）を作成して，極力手作業を省いている．

(a) 変数名変遷表



(b) 変数名対照表

図 9．系列化の手順

5.4. 手入力との比較

最後に，手入力との比較をしておく．手入力の場合は，専門業者に外注することが多いだろう．費用はおおよそ，セル当たり単価に入力する総セル数を乗じたものになる．外注する場合は，(a) テキストデータ化する統計表の指定，(c) 出力フォーマットの作成・指定，(d) 校正方法等の仕様を固めて指示する．校正は一般に，2 名が独立に入力したものを突合して

誤入力を検証する（と謳っている）．ただし，これにはオプション料金がかかったり，誤入力が残っているケースも散見されるため，品質管理が必要である．以上の仕様を業者と合意できれば，作業管理は委託できる．また，入力する作業員を増やせば並列処理ができるため，納期も短縮できる可能性がある．

ocrstats を使って内製する場合，費用としては，手作業の工程の人件費，管理費，OCR 利用料がかかる．手作業の工程はアノテーションと校正である．前述のとおり，アノテーションはさほど時間はかからず，集中的に実施できる．もっとも時間がかかるのが校正で，表や作業員によって 1 表あたり 10〜30 分程度かかる．さらに作業全体の進捗管理も必要となろう．校正は一定のトレーニングと慣れが必要であることから，容易には作業員を増やせず並列処理が難しい．このため納期が長くなる．OCR 利用料は OCR 処理を期間を区切って集中的に実施すれば，大きな費用負担にはならないだろう [19]．

以上，大まかな試算では，ocrstats を利用することで費用としては手入力の 1/2〜3/4 程度に抑えられる．ただし，進捗管理が必要なこと，納期がかかることが欠点である．

6. 課題と今後の展望

本稿では，OCR を使った統計表画像のテキストデータ化の要件やノウハウを解説し，支援ツール ocrstats の機能を紹介した．締めくくりに，課題と今後の展望を整理する．

ocrstats の課題として，優先順位順に以下の機能強化が挙げられる．

1. 校正支援．校正は，最終的には問題がありそうなセルを目視でチェックする必要がある．この問題セルの特定が最大のボトルネックである．現在，検証用画像によって，原本画像で確認すべき箇所の特定は容易になっている．校正用の Excel ファイルにも同様にセルを強調表示することで，校正作業の効率を改善できるだろう．

2. 複数の出力結果の統合による差分の利用. 異なる OCR エンジンを使って，OCR 処理と表復元をおこない，復元結果を突合する [20]．結果が相違するセルは，誤認識の確率が高いため，強調表示して，目視でチェックする．

3. レイアウト解析の自動化. レイアウト解析の技術は急激に進展しており，さまざまなアルゴリズムやパッケージが公開されている [21]．これらを活用できれば，レイアウト解析や

[19] OCR 利用料は，従量制や月額制の設定が多い．CLOVA OCR は 1 ヶ月 2.2 万円の月額制である．OCR 処理を集中的に実施すれば，費用はこれで収まる．従量制の場合も 1 枚当たりの単価は低い．例えば，Amazon Textract であれば，約 6,500 枚（表）の画像の OCR 処理費用はせいぜい 20 米ドル程度ですむ．

[20] 同様の発想で文章資料の OCR 処理の正確性を高めているものとして Bleamer (2018).

[21] 深層学習を利用した表検出・表レイアウト解析には，Ohta et al. (2019), Prasad et al. (2020), Raja, Mondal, and Jawahar (2020) などがある．深層学習による一般的な文書画像解析を，日本語歴史資料を含めた資料に

アノテーションも含めて，校正までの過程は全自動化できる．また，表側も含めてテキストデータ化できるようになる（4.7 節）．

4. 見開き位置の自動検出とページ分割. これはすでに深層学習等を利用した Python スクリプトのソースコードが公開されている [22].

5. 表頭の解析と OCR の実装.

6. 統計表画像データセットの学習による OCR の改善. 解像度が低い原画像や，漢字・かな・カナは旧字を含むこともあり，誤認識が散見される．戦前の旧字を含む活字のデータセットを学習させることが考えられる．また，認識範囲不良も，統計表画像を学習させることで，改善が見込めるかもしれない．

（政府）統計表のテキストデータ化全般については，テキストデータ化する統計の拡大（各種政府統計や「府県統計書」など），テキストデータおよび作業情報のレポジトリの作成（情報の集約化），「みんなで翻刻」[23]のようなクラウドソーシング型プロジェクトの展開，などが挙げられる．特に，府県統計書のような大規模な統計は，多数の市民の参加と協力を募って校正作業を進め，公共財をつくり上げることが考えられる．

引用文献

青池亨・木下貴文・里見航・川島隆徳（2019）「機械学習のための資料レイアウトデータセットの構築と公開」『じんもんこん 2019 論文集』，pp. 115-120.

青池亨・里見航・川島隆徳（2018）「資料画像中の挿絵領域の自動抽出及び画像検索システムの実装」『じんもんこん 2018 論文集』，pp. 97−102.

五十嵐彰（2020）「『日本帝国統計年鑑』を電子化しての雑感」『社会と統計（立教大学社会情報教育研究センター研究紀要）』6, pp. 55-64.

北本朝展・カラーヌワット　タリン・Alex Lamb・Mikel Bober-Irizar (2019) 「くずし字認識のための Kaggle 機械学習コンペティションの経過と成果」『じんもんこん 2019 論文集』，pp. 223-230.

総務省統計局（2020）「統計表における機械判読可能なデータ作成に関する表記方法」(https://www.soumu.go.jp/main_content/000723697.pdf)

活用したツールとしては，LayoutParser (Shen et al. 2021)や青池・里見・川島（2018）などを参照．アノテーション済みの日本語歴史資料レイアウトデータセットとしては，「古典籍資料」や「明治期以降刊行資料」が「NDL-DocL」（青池・木下・里見・川島, 2019）として，『人事興信録』（第 17 版）（1953 年）のデータセットが「HJDataset」(Shen, Zhang, and Dell, 2020) として公開されている．

[22] https://github.com/ndl-lab/ssd_keras

[23] https://honkoku.org/

山崎翔平（2018）「歴史研究におけるデータ利活用事例：近代日本における府県別平均就学年数の推計」mimeo. (URL: http://hdl.handle.net/2261/00076476)

Bleemer, Zachary (2018) “The UC Cliometric History Project and formatted Optical Character Recognition,” Research & Occasional Paper Series: CSHE.3.18, UC Berkeley. (URL: https://cshe.berkeley.edu/publications/uc-cliometric-history-project-and-formatted-optical-character-recognition-zachary)

Ohta, M., Yamada, R., Kanazawa, T., and Takasu, A. (2019) “A Cell-detection-based Table-structure Recognition Method”. In Proceedings of the ACM Symposium on Document Engineering 2019 (DocEng '19). Association for Computing Machinery, New York, NY, USA, Article 27, 1–4. DOI: https://doi.org/10.1145/3342558.3345412

Prasad, D., Gadpal, A., Kapadni, K., Visave, M., and Sultanpure, K. (2020). “CascadeTabNet: An approach for end to end table detection and structure recognition from image-based documents.” In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 572-573). arXiv preprint arXiv:2004.12629v2.

Raja, S., Mondal, A., and Jawahar, C. V. (2020). “Table Structure Recognition using Top-Down and Bottom-Up Cues”. In European Conference on Computer Vision (pp. 70-86). Springer, Cham. arxiv preprint arXiv:2010.04565v1.

Shen, Z., Zhang, R., Dell, M. (2020) “A Large Dataset of Historical Japanese Documents with Complex Layouts”. arXiv preprint arXiv:2004.08686v1.

Shen, Z., Zhang, R., Dell, M., Lee, B. C. G., Carlson, J., and Li, W. (2021). “LayoutParser: A Unified Toolkit for Deep Learning Based Document Image Analysis”. arXiv preprint arXiv:2103.15348.

Shen, Z., Zhao, J., Dell, M., Yu, Y., and Li, W. (2021) “OLALA: Object-Level Active Learning for Efficient Document Layout Annotation,” arXiv preprint arXiv:2010.01762v3.