地方公共団体のオープンデータ取組支援ツールの実現可能性調査

調査報告書 (概要版)

tonoi株式会社

本調査の概要・目的・調査範囲

目的

- 地方公共団体において自治体標準オープンデータセット（以下「自治体標準ODS」）取組の支援となるツールの可能性の調査を目的とする
- 印刷を前提とした複雑なレイアウトのデジタル資料を、機械判読可能なデータへ整形するにはコストがかかる
- 生成AIや従来の機械学習等のツールの支援により、複雑なレイアウトのデータを機械判読可能なデータに何らか自動的に整形し、作業簡略化につながる可能性がある

▽

生成AI等を使ったデータ取り組み
支援ツールのプロトタイプを検証

調査範囲

① 現状分析

- 自治体標準ODSを作成するために、地方公共団体がどのようなデータを保有しているか、３か所でのヒアリングに参加してその結果をまとめ、データ変換時に問題になりそうな点も合わせて報告する

② 技術調査

- 生成 AI を含む、データ変換に利用できそうなツールについてまとめる

③ 適用可能性評価

- 地方公共団体から提供されたデータをもとに自治体標準ODSに変換できるか調査し評価すべきツール・サービスを選定する

④ プロトタイプの開発

- 地方公共団体の持つ元データを変換するプロトタイプを開発する

⑤ 地方公共団体での実証

- 開発したプロトタイプを3か所の地方公共団体で実証実験し可能性を調査する

※本調査研究は、令和5年12月18日から令和6年2月29日にかけて実施したものであり、生成AIの分野は急速に進化していることから、最新の技術情報とは異なる可能性があることに留意していただきたい。

① 現状分析

- ファイル形式
  - 原課でデータを作成・変換する場合はスプレッドシートが中心
  - データを他の課から入手する場合は、データがPDFであることがある
  - PDFの中には、過去の紙の資料をスキャンして保存しているものがある
  - 一部、Just Systemsのファイルが存在している
- 表の複雑さ
  - 原課の扱うデータは紙への印刷を想定したデータが多い
  - 議会用資料などでは、縦組みと横組みが1つの表に混載されている
  - 100ページ以上ある資料もあり、表データがその途中に出てくることもある
  - データ内容の過不足やバラツキ等
    - 「1丁目2番」や「 1-2」など住所表記の揺らぎがある
    - 都道府県名や市町村名が省略されている住所がある
- データ項目の不足
  - 自治体標準ODSに必要な項目が不足している
  - 住所が入っていない元データも存在している
- 利用環境
  - LGWANの制限によりWebツールによるデータ変換はインターネット用端末にUSBメモリでデータを受け渡して対応していることもある
  - 利用PCへのソフトウェアのインストールは可能
  - ファイルフォーマットだけではなく、利用ソフトウェア、認証などのIT基盤が、地方公共団体・担当課ごとに異なっている
- 組織的状況
  - オープンデータの目的や意義の理解が広がっておらず、組織としてオープンデータの取り組みの優先度が上がらない
  - 元データに機密情報等が含まれる可能性がある

地方公共団体ごとの自治体標準ODSへの取り組み

比較項目	小規模基礎自治体	中規模基礎自治体	都道府県
元データの発生源	情報システム部門職員の手作業	情報システム部門が原課から入手	各市町村が作成
データ整形の担当	情報システム部門職員の手作業	外部委託	各市町村が整形
データの公開サイトへの登録	情報システム部門職員の手作業	外部委託	情報システム部門が登録
公開データの管理	情報システム部門職員の手作業	外部委託	専用CMS
データ公開前のチェック担当	原課	原課	各市町村
公開場所	地方公共団体サイト内	専用サイト(CKAN)	専用サイト(Shirasagi)

- 都道府県においては、各原課・市町村がそれぞれ自治体標準ODSを内製している。情報システム部門はそれを取りまとめて専用CMSに登録するのみでありワークフローの効率化がなされている
  - このワークフローを実現するために各市町村ごとにCMSのアカウントを作成して配布した
- 基礎自治体ではデータの調達・作成において、以下のような作業負担がみられた
  - 消火栓の位置情報は紙の地図でおおよその位置を見ながら職員が手作業で取得
  - 施設情報はそれぞれの担当職員が施設のWebサイトを見て手作業でデータを作成
  - 住所の揺らぎの修正は手作業で対応 (1丁目2番 → 1-2 など)

② 技術調査

2024年1月調査

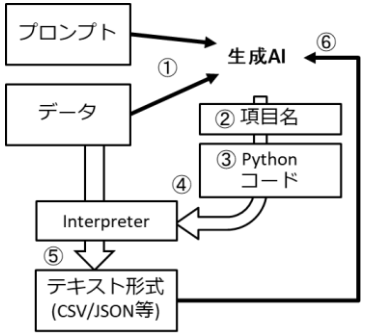
データの形式変換で使われているTableau Prep Bulider と、5種類の生成AIを調査対象として、３つの項目で技術調査を行った。

- 多様なファイル形式：スプレッドシート、PDF、CSV（JIS、SJIS、EUC、UTF-8）、外字の読み込み
- レイアウトの整形：複数の表、グラフの混在、セル結合など
- 地方公共団体での利用可能性：学習させない選択肢の有無など

	調査項目	Tableau Prep	Azure + GPT4	ChatGPT	Claude2.1	Gemini Pro	LLaMA
技術調査	多様なファイル形式	△	○	○			
	レイアウトの整形	○	○	○			
	地方公共団体での利用可能性		○		○	△	△

スプレッドシート・PDFに対応し、整形・データ補完が可能で、
閉鎖環境に対応した Azure + GPT4 をプロトタイプ開発環境として選定

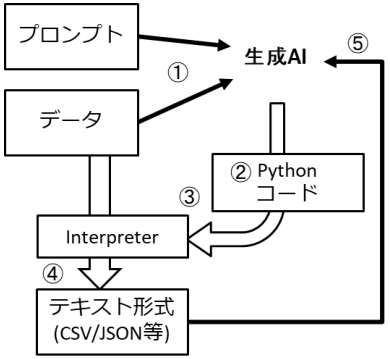
技術背景: 見いだされた生成AIによるデータ読み込みの違い



スプレッドシート

1. プロンプトを通してデータを生成AIに渡す
2. 生成AIがデータを読み取って項目名を抽出
3. 項目名をもとに生成AIがデータ読み取りPythonコードを作成
4. 作成されたPythonコードをInterpreterで実行
5. Interpreterがデータを読み取ってテキスト形式で出力
6. 出力されたテキスト形式を生成AIが読み取る

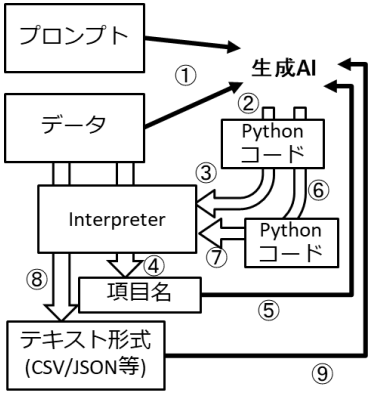
生成AIが元データのレイアウトを把握しているため、表などが再現されやすい



画像型PDF

1. プロンプトを通してデータを生成AIに渡す。画像型なのでPyPDFライブラリを指定
2. 生成AIが画像型PDFを読み取るPyPDFライブラリを使ったPythonコードを作成
3. 作成されたPythonコードをInterpreterで実行
4. Interpreterがデータを読み取って画像認識をしたテキストを出力
5. 出力されたテキストを生成AIが読み取る

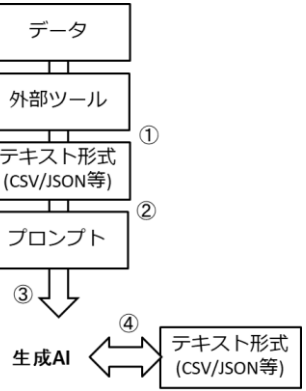
生成AIはPDFを読めず、PyPDFでは画像認識によるテキスト抽出だけなので、データの項目名をレイアウトから抽出することができない
CSV構造の再現は読み取ったテキストだけから推論するため表などのレイアウト再現は難しい



テキスト型PDF

1. プロンプトを通してデータを生成AIに渡す
2. 生成AIがデータを直接読めないと判断して、PDFを読み取るTabulaライブラリを使ったPythonコードを作成
3. 作成されたPythonコードをInterpreterで実行
4. Interpreterがデータを読み取って項目名を出力
5. 出力された項目名を生成AIが読み取る
6. 項目名に応じてデータを読み取るPythonコードを生成AIが作成
7. 作成されたPythonコードをInterpreterで実行
8. Interpreterがデータを読み取ってテキスト形式で出力
9. 出力されたテキスト形式を生成AIが読み取る

PDF表構造読み取りはTabulaライブラリの性能に依存



プロンプト埋め込み

1. データを外部ツールでテキスト形式に変換
2. プロンプトにテキスト形式データを埋め込む
3. データを含むプロンプトを生成AIに入力
4. 以後、プロンプトを通してデータを整形

生成AIがテキスト形式以外の読み取りに対応していなかった場合、Pythonのpands/tabula/PyPDFなど外部ツールを使ってデータをテキスト形式のCSVにしてプロンプトの中に埋め込む。
本項に該当する生成AIとしてClaude2.1とGemini Proが上げられるが、対応トークン数が20万、100万と非常に大きく、テキスト形式であれば780Kbyteに相当し、多くのデータをプロンプトに埋め込むことができる。

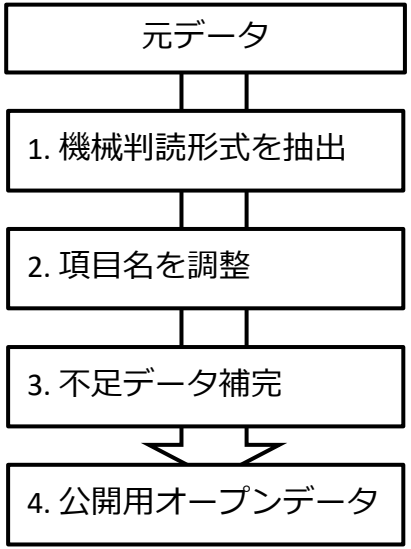
③ 適用可能性評価

Azure + GPT4を用いて、以下の評価項目で機能評価を行い、対応できることを確認した。

評価項目		評価内容	評価
ファイル結合	複数ファイル	複数のファイルを結合できるか	○
	主キー参照	主キーをもとに同一行に補完できるか	○
	複数キー参照	複数キーをもとに同一行に補完できるか	○
	項目名の自動振り分け	住所/アドレスなど揺らぎのある項目名を結合できるか 違う項目名を統合できるか	○
テキスト処理	テキスト分割	東京都千代田区などを都と区に分離できるか	○
	テキスト結合	東京都、千代田区を東京都千代田区に結合できるか	○
	表記ゆれ検出	東京都、Tokyoなどの表記揺れを検出できるか	○
	半角・全角	半角・全角の表記揺れを検出できるか	○
	表記ゆれ修正	検出された表記揺れを修正できるか	○
	読み仮名	カナ・英字の読み仮名を作れるか	○
データ補完	不足項目検出	項目名で不足分を割り出せるか	○
	全国地方公共団体コード	補完データ取得可能か	○
	町字ID	補完データ取得可能か	○
	緯度経度	団体名・住所から緯度経度を割り出せるか	○
	郵便番号	住所から郵便番号を探せるか	○

プロンプトを用いた自治体標準ODSへの変換

ご協力いただいた地方公共団体より提供を受けた元データを自治体標準ODSにデータ変換する調査を行ったところ、効率の良い手順と有効なプロンプトが見いだされた。



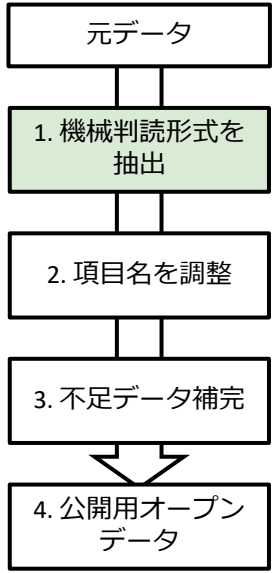
プロトタイプを用いたデータ変換の手順

1. 元データから機械判読可能なデータを抽出
   - スプレッドシート、CSV (含むZIP) は高精度で生成AIから読み込める
   - PDFはテキスト抽出はできるが表の再現は限界がある
   - ページ数の多いPDFは不要ページの読み飛ばしに時間と課金が掛かるため
     事前にAdobe Acrobatなどを使って整形対象のページだけを別ファイルに切り出し
   - 生成AIで読み込みが難しいPDFは他のツールでCSVに事前変換
2. 項目名を自治体標準ODSに合わせ調整
   - 項目名が揃っていれば、後の工程で同じプロンプトが利用可能
3. 不足データを補完
   - 安全な閉鎖環境内で補完データを提供するAPI (Proxy-API)を一部利用
   - 使用条件によっては外部APIやDBからも不足データを補完
   - 住所の先頭に都道府県を付けるなどは生成AIのプロンプトで対応
4. 完成したデータをダウンロードして保存
   - ダウンロードしたCSVをExcelで確認するためには、テキストエディタなどでCSVを開いてBOM付きUTF-8でエンコードして保存

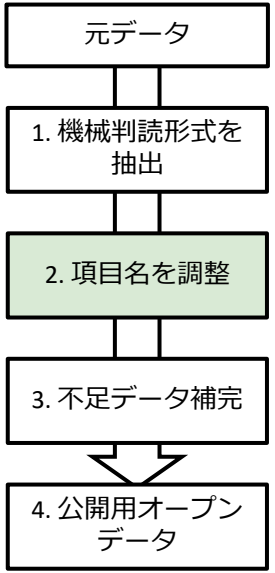
機械判読可能なデータの抽出プロンプト

スプレッドシートやPDFから表形式のデータを生成AIが抽出するプロンプト

- 「入力ファイルをCSVに変換して保存してください」
  - 多くのスプレッドシートに対応可能、一部の簡単なPDFにも対応
  - プロンプトは毎回同じ処理をしないため、読み込みエラーが出てもあきらめずにプロンプトで対話を続ける
  - 複数のプロンプトは組み合わせても生成AIは認識するが、あまりに多くの指示を一度に行うと失敗する確率が高くなる
- 「最初のページはタイトルなので飛ばし２ページ目以降にある表をCSVに変換して保存してください」
  - タイトルがあるスプレッドシートは読み飛ばすことができる。複数枚シートがあるスプレッドシートもシート名を指定することで読み込まさせられる
- 「もう一度やってください」
  - 生成AIは言葉の関連性が高い処理方法を行うので、エラーが出てもプロンプトでもう一度繰り返せば２番目に関連性の高い処理方法を行う
- 「実行してください」
  - 生成AIはデータを読み込むPythonコードを生成だけして実行しないことがある。プロンプトで実行許可を出すと実行してデータを読み込む
- 「入力ファイルの中にある表の1行目を項目名としてCSVに変換して保存してください」
  - 多くは項目名を自動認識するが、失敗した場合は１行目を項目名を使うように指示
  - 所在地という項目名を住所と勝手に考えて「住所がありません」と返すなど項目名に関してはハルシネーションが発生する。不正確な項目名が発生した場合は「項目名は”番号”、”施設名”、”所在地”、”分類”」などとプロンプトの中で明示
  - 項目名がセル結合されていても読み込めるが、失敗するときは上の手法を試す



項目名の調整プロンプト



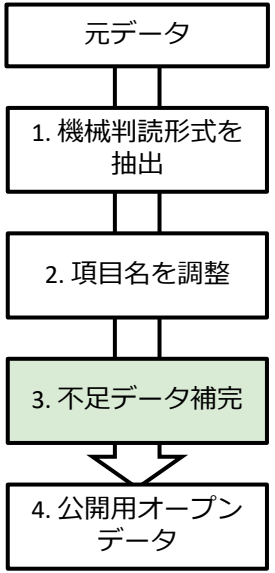
自治体標準ODS形式に項目名を調整するプロンプト

- 「
入力したCSVファイルを自治体標準オープンデータセットに合わせて整形します。 下記の自治体標準オープンデータセットの項目名に合うようにデータを配置してください。作成するCSVは下記の項目名をすべて含み、データないところは空白を入れてください。作成したCSVファイルを保存してください。
自治体標準オープンデータセットの項目名:
{ 項目名, 全国地方公共団体コード, ID, 地方公共団体名, 名称, 名称_カナ, 名称_英字, 医療機関の種類, 所在地_全国地方公共団体コード, 町字ID, 所在地_連結表記, 所在地_都道府県, 所在地_市区町村, 所在地_町字, 所在地_番地以下, 建物名等(方書), 緯度, 経度, 高度の種別, 高度の値, 電話番号, 内線番号, FAX番号, 連絡先メールアドレス, 連絡先FormURL, 連絡先備考（その他、SNSなど）, 郵便番号, 法人番号, 法人の名称, 医療機関コード, 診療曜日, 診療開始時間, 診療終了時間, 診療日時特記事項, 時間外における対応, 診療科目, 病床数, 災害拠点分類, 状況, URL, 画像, 画像_ライセンス, 備考 }
」
  - 自治体標準ODSの項目名をデジタル庁のWebサイトから自動認識させることはうまくいかない。項目名の明示が必要
  - #1で読み込んだ機械判読可能なデータの項目名と自治体標準ODSの自動適合はたまに成功するが、住所＝所在地の表記の揺らぎなどで読み込めないことも多い
  - 項目名の調整は生成AIが作成したPythonコードで行われる
- 「
元のデータと自治体標準オープンデータセットの項目の適合関係:
["施設名称" = "名称", "開設種別" = NaN, "郵便番号" = "郵便番号", "所在地" = "所在地_連結表記", "電話番号" = "電話番号", "FAX番号" = "FAX番号", "開設年月日" = NaN]
」
  - 項目名の自動適合に失敗した場合は、プロンプトで細かく修正するよりも適合関係を明示的に指定するほうが所要時間が短い

不足データ補完プロンプト

自治体標準ODS形式に必要なデータを補完するプロンプト

- 「住所の先頭に”福岡県”と入っていないデータには“福岡県“をいれてください」
  - 住所の都道府県に関する表記の揺らぎを一度に直す
- 「続いて、不足しているデータをAPIを使って補完します。「名称_英字」のデータを「名称」から作成します。以下のURLの　%名称%のところにURLエンコーディングした「名称」のデータを入れてAPIを使用してください。 APIの戻り値からデータを抜きだし、既存の項目「名称_英字」に入れてください。CSVファイルで保存してください。https://proxy-api-gss.azurewebsites.net/japanese-to-english?jp=%名称%」
  - 閉鎖環境内で提供した日本語の英字よみを作成するAPIで“名称_英字”を作成
  - 次のOSSを利用した (GitHub - dx-junkyard/OpenData-Bridge-proxy-api)
- 「
続いて、所在地に関わるデータ補完をします。項目「所在地_連結表記」のデータを引数にして下記のURLで%住所% のところをURLエンコーディングした「所在地_連結表記」のデータを入れてAPIを使用してください。
https://proxy-api-gss.azurewebsites.net/digital-geocode?address=%住所%
戻り値と項目の対応は（戻り値　項目）（city　所在地_市区町村）（fulladdress　所在地_連結表記）（lat　緯度）（lg_code 全国地方公共団体コード）（lon 経度）（prefecture 所在地_都道府県）（town 所在地_町字）（town_id　町字ID）です。APIの戻り値からデータを抜きだし、既存の該当項目にデータを入れてください。CSVファイルで保存してください。
」
  - デジタル庁のベースレジストリGeocoderを利用して住所から緯度経度・全国地方公共団体コード・町字ID・所在地_連結表記・所在地_都道府県・所在地_市区町村・所在地_町字を取得
    GitHub - digital-go-jp/abr-geocoder: Address Base Registry Geocoder by Japan Digital Agency
  - 正しい住所でなければ間違ったデータが入ってしまう
  - 住所によってはGeocoderでデータを見つけられないことがある



公開用オープンデータの確認

生成AIの支援により作成されたオープンデータの確認

- CSVファイル形式の確認
  - BOMなしUTF-8で作成されるため、Excelで開くと文字化けする。Windows付属のメモ帳などテキストエディタを使用してBOM付きのUTF-8で保存しなおすと文字化けが解消される。
  - Pythonの日本語対応の問題でCSVデータの途中で改行が入ってしまいCSVファイル読み込みでエラーとなることがある。
- データの正確性の確認
  - 生成AIが項目名を間違えたPythonコードを生成してしまうことがある。「緯度」が「練度」になっていた。項目名の目視確認が望ましい。
  - APIを利用してデータ補完した場合、見つからなかった項目にNaNや空白が入る。空欄のデータが許容されない場合は手動でデータ補完する必要がある。
  - 元データが誤っていたり、APIが間違ったデータを補完することもある。
- ライセンスの確認
  - 生成AI経由でAPIを実行した場合に、ハルシネーションにより、異なるAPIが呼ばれるケースがあった。これは、チャットの内容に合わせてより一般的なAPI呼び出しに最適化しようとしたと推測されるが、APIにはそれぞれ利用規約があるため、利用したAPIで補完したデータがオープンデータとして公開可能か確認が必要になる。

④プロトタイプの開発

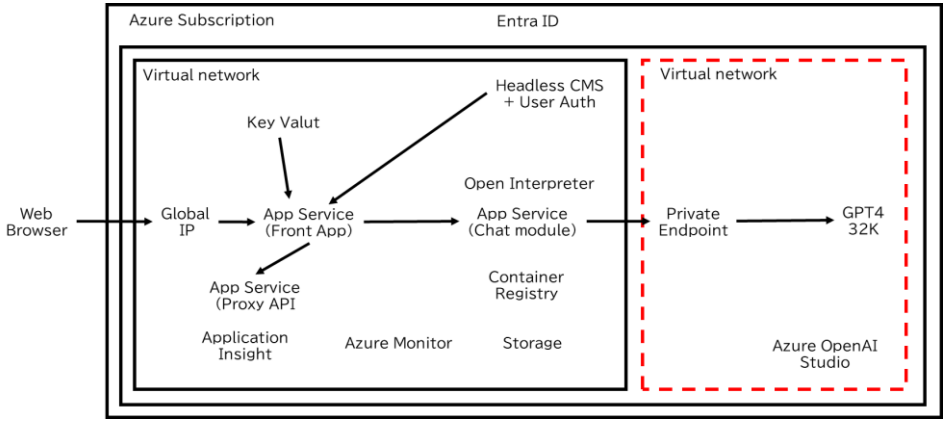
2024年1月開発

※ 選定した技術の適用可能性調査を行うため、既存ossを利用してプロトタイプ開発を実施

データ整形画面

プロンプトを介して生成AIを利用して
データ整形するためのチャット画面

プロトタイプ構造



※Azure OpenAIのモデルにはgpt-4-32k Version 0613を利用

閉鎖環境の生成AIを用いた
独自Webアプリケーションを構築

注：生成AIの実行結果をpythonスクリプトに書き出し、変換に成功した時の一連のスクリプトを保存し再利用するコンセプトのOSSを利用したが、今回の調査期間内では再利用するに至らなかった

⑤ 地方公共団体での実証

- ヒアリング調査にご協力いただいた3つの地方公共団体でプロトタイプを使った実証実験を行った
- ツールの利用が難しかったケース
  - 生成AIは導入検討時に少し利用した程度
  - 事前に提供したプロンプトを入力して試すことだけ実施
  - エラーが出たときはそれ以上の対応が難しかった
- ツールが十分に使えたケース
  - 常に新しい技術を自分で試している
  - プロトタイプのマニュアルにあることだけではなく、自分で創意工夫をして様々な可能性を試していた
  - 全般的に好印象であり、印刷前提のファイルから機械判読可能なデータを抽出できる機能だけでも使いたいという意見があった
  - セキュリティで制限されていないバージョンを作ってほしいという要望があった

	生成AI未経験者	生成AI経験者
データ整形の手間は削減できるか	利用者を選ぶ	削減につながる
操作性	難しい	直感的に使える
良かった点（全6項目※）	3	6
データ作成できたか	できなかった	できた
整形・補完の正確性	半分以上が不適切	期待通り
他の業務でも使うか	思わない	思う

※ PDF読み込み、フォーマット変換、表構造抜き出し、項目の並び替え、データの補完、重複データの削除の6項目

本調査を通したまとめ

地方公共団体のデータ公開支援にあたり、現状の課題についてヒアリングを行った。その結果、元データが印刷用レイアウトであることが、データ変換作業を難しくしている事が分かった。また自治体標準ODSの作成時には、住所などの元データが欠けていることや、紙やWebサイトを参照して作成するなどの作業負荷も課題になっている事が確認できた。技術調査では、複数の生成AI等を試した結果、機密情報を取り扱う可能性を想定して、今回はAzureとGPT-4の組み合わせを選定した。適用可能性評価では、自治体標準ODSを作成するための生成AIのプロンプトを明らかにし、一定の手順を用いて成功を収めるも、元データがPDFだった場合の表構造の認識や、作成したデータの正確性には課題が残っている。

試作したプロトタイプを地方公共団体職員に使用してもらった結果、生成AIに慣れた職員は肯定的な反応を示す一方で、技術に不慣れな職員からは操作性の改善要望が寄せられた。成功したケースではデータ抽出とデータ補完について高い評価が得られた。生成AI技術の進化に伴い、直接的なデータ抽出が可能になりつつあり、今後の自治体標準ODSへの変換支援ツールの実用性の向上が期待される。

ヒアリング	技術調査	適用可能性評価	プロトタイプ	実証実験
•自治体標準ODSへのデータ変換にコストがかかっている •印刷を想定した元データが多い •項目が不足している元データが多い •元データがPDFであることもある	•機密情報対応と、スプレッドシート・PDF読み取りを両立するツールは少ない •PDF内のテキストが画像形式であると読み取りが困難 •PDFからの表構造の認識には問題がある	•生成AIで自治体標準ODSに変換するために、効率の良いプロンプトがあることがわかった •作成した自治体標準ODSは間違っていることもあるため、目視での確認が必要	•プロンプトによるチャットを通してデータ整形が可能 •閉鎖環境によって機密情報の取扱が可能 •自治体標準ODS向けのデータ補完が可能	•生成AIに慣れた職員ならばデータ変換支援ツールとして有効活用可能 •生成AIに不慣れた職員向けにはもっと簡単な操作が必要 •データ抽出とデータ補完が有用であった