OCR とラベリングによる書類整理自動化システムの構築と有用性の評価

青山学院大学理工学部
情報テクノロジー学科Dürst研究室

学籍番号：15820094

嘉松 一汰

目 次

第1章　はじめに

1.1　研究背景

日本企業の RPA (Robotics Process Automation [7]) 導入率は全体で 38%, 中小企業では 25%となっており, 非常に少ないことがわかる (図 1.1). また, 大企業と中小企業の間に 20%以上の差があり, 技術や規模による格差も見て取れる. これらの原因となっており要因として考えられることは, 大きく分けて 2 つある.1 つ目は,RPA には専門領域と非専門領域が存在するということである. 専門領域は PC 上の操作や, デジタルの領域における処理である. いっぽうで, 紙媒体の処理等の, アナログの世界で行われる処理は非専門領域としている. 特に, 手書きの文字や画像の認識を高い精度を保ちながら自動で行うことは, 現代では非常に困難なことである. 縦書き文字横書き文字が混在していたり, 旧字体や特殊文字等の組み合わせも考えられるため. 例外的な処理までを自動で行う必要があるからである [1]. 2 つ目の原因は, 紙媒体の業務を行っているコミュニティの IT 知識の乏しさにある. 詳しくは次のセクションで説明する. このような現状を踏まえて, 次章からは,OCR 技術を使用したアプローチを提案する.

図 1.1: 企業の RPA 導入率

1.2 研究目的

本研究の目的は,IT 知識が乏しく, 紙媒体の業務を行っているコミュニティを中心に, RPA を使用して紙媒体の処理を自動で行うシステムを作成することである. 具体的な内容に入る前に, 紙媒体の業務と IT 知識の関連性について深堀りする. 日本で働く人事・総務担当者に,「紙媒体中心の業務で不便を感じたことがあるか？」とアンケートを取ったところ,61%が不便を感じたことがあると回答した (図 1.2). 上記の理由として, システム障害への恐怖感や,IT 知識の乏しさが挙げられます. しかし, 一連の流れを RPA にすることで,IT 知識の有無に関わらず, システムを運用することを目指す.

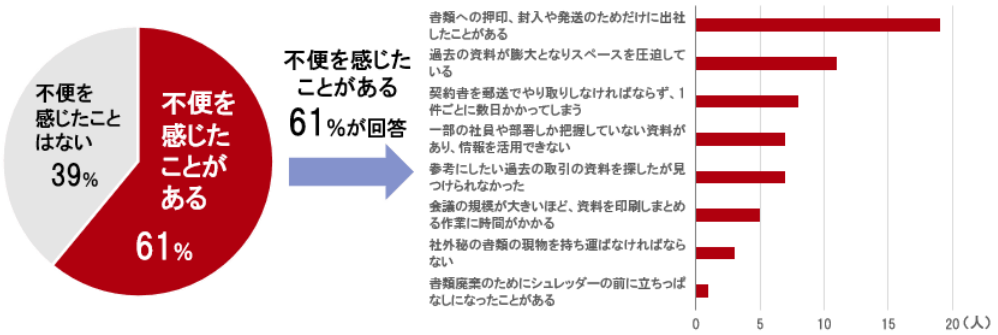


図 1.2: 紙媒体中心業務に関するアンケート

1.3 構成

本論文は 6 章構成になっている. 第 1 章では, 本研究の背景や目的を述べる. 第 2 章では既存手法との比較や課題を述べる. 本研究での実装手法を第 3 章で提案し, 第 4 章で実験と評価, 第 5 章で結果の考察を述べ, 第 6 章でに本研究の総括を行う.

第2章　関連研究

2.1　基礎技術

2.1.1　Ruby

Ruby [10] [5] は,1993 年にまつもとゆきひろ (松本行弘，通称 Matz) が日本で開発したオブジェクト指向言語である．名前はまつもとの同僚の誕生石であるルビーが由来となっている. 本研究では, システムのバックエンドでのテーブル操作に使用する.

2.1.2　Ruby on Rails

デンマークのプログラマであるデイヴィッド・ハイネマイヤー・ハンソン（通称 DHH）氏によって作られた．エンジニアの間では略称である Rails または RoR と呼ばれることも多く, 簡単なコードで Web アプリケーションの開発ができるように設計されている [9]．本研究では, フロントエンドの ERB ファイルを始めとした,Web システムとしての実装に使用する.

2.1.3　Google Cloud Vision API

Google が提供する画像認識サービスのことである.Google 独自の機械学習モデルを採用しており, 効率的に画像を分析し, オブジェクトや顔の検出や手書き文字の読み取り, 有用な画像メタデータの構築など様々なことを実現できる [6]．本研究では, ドキュメントの画像に対して OCR を行い, 単語ごとに分類するために使用する.

2.1.4　WordNet

大規模な語彙データベースのこと. 名詞, 動詞, 形容詞, 副詞は, 認知同義語 (Synset) のセットにグループ化され, それぞれが異なる概念を表現する．概念は, 概念的意

味的および語彙的関係によって相互にリンクされている. テーブル構造としては, 概念テーブル (Sense), 語彙テーブル (Word) とそのリレーションテーブル (Synset) によって形成されている [4]. 本研究では, 単語を該当するカテゴリに分け, ラベリングを行うために使用する.

2.2 クラスタリングの既存手法

2.2.1 非階層的クラスタリング

目的のデータを事前に定義されたクラスタ数に分解することによって行われるクラスタリング方式のことである. 代表的な手法として, クラスタ内の分散を最小化するようにデータポイントをグループ化する k-means アルゴリズムが挙げられる [3].

2.2.2 階層的クラスタリング

データポイントを機構造の階層に分割して行うクラスタリング方式のことである. 代表的な手法には大きく分けて, 凝集型と分割型の 2 種類がある. 凝集型は, 木構造の下から上へクラスタを統合していく方法である. 対して分割型は, 上から下へクラスタを分割していく方法である. データの類似度に基づいて徐々にクラスタを形成していくため, クラスタ数を事前に決定する必要がないため, クラスタの適切な数が不明瞭な場合や, データの階層的な構造を理解したい場合には分割型が有用である [2].

2.3 クラスタリングの課題

クラスタリングの大きな課題として挙げられるのは, 適切なクラスタ数の決定である. 特に非階層的アルゴリズムの場合は, クラスタの数を事前に定義する必要があるが, これが直感的に明らかでない場合が非常に多い. クラスタリングは, データの中から自動的にパターンや構造を見つけ出す, 教師なし学習 [8] と呼ばれる手法を使用しているため, 生成されたクラスタリングの解釈が主観的になりやすい. 言い換えれば, クラスタリングの結果をどう捉えるかがこちらに委ねられている.

第3章　提案手法

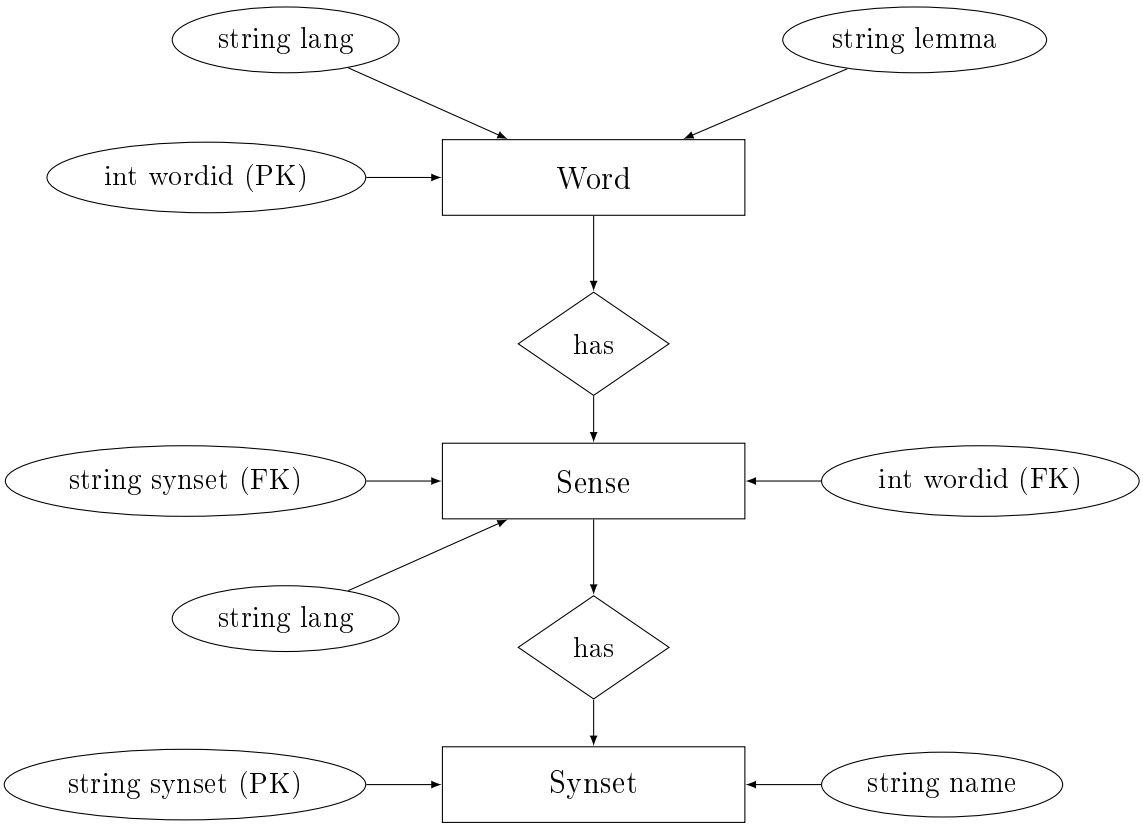
本研究で活用する手法, それに対するアプローチをはじめ, テーブルの構造や内容などのシステム設計を示す. また, WordNet データベースとの連携方法に加えて, ドキュメントをどのようにカテゴリ分けするかを具体的に示す.

3.1　システム設計

本研究のシステムのバックエンドに当たる部分を具体的に示す.

3.1.1　データベース設計

本研究で使用するテーブルには 3 種類ある. 単語を格納する Word テーブル, 単語の定義を格納する Synset テーブル, 前述した 2 つのテーブルの中間テーブルとなる Sense テーブルである. 以下に, これらを表す ER 図を示す



3.1.2 モデル設計

Word モデル　一対多の関係で, 複数の Sense モデルへのリレーションを持っており, Sense モデルを経由して, Synset モデルへのリレーションを辿ることで, 単語からその定義を取得する. 場合に応じて適切な言語のレコードを活用するため, 言語での絞り込みを行うメソッドを持つ.

Synset モデル　一対多の関係で, 複数の Word モデルへのリレーションを持っており, Sense モデルを経由して, Word モデルへのリレーションを辿ることで, 特定の定義をもつ単語を取得する. Word モデルと同様に, 言語での絞り込みを行うメソッドを持つ.

Sense モデル　Sense モデル, Synset モデル両社と一対多のリレーションを持っており, 二つのモデルの中間テーブルの役割を担う. 単語から定義を取得する処理は, Synset テーブルを経由して行う.

3.2 提案する手法 ・ アプローチ

本研究での手法の流れを以下に示す. 始めに, カテゴライズしたいドキュメントファイルをシステムにアップロードする. その後, ファイルに対して Google Vision API を使用した OCR を行い, 内容の単語群を取得する. OCR と親和性のある代表的なプログラミング言語として Python が挙げられるが, http リクエストによって言語を問わず簡単に利用できる点や, OCR 処理に対するオプションの豊富さから, 本研究で使用する. その後, WornNet データベースと連携し, 単語ごとの定義を取得する. WordNet は日本語, 英語双方に対応した対規模な語彙データベースであるため, 言語の入り混じったドキュメントに対しても問題なく処理を行うことができるため, 対応可能な語彙の量と汎用性を加味し, 本研究で使用する. 各々の定義を元に単語がどのカテゴリに属するかを判別し, 単語ごとにラベリングを行う. ラベリング結果から, 最終的なドキュメントのカテゴリを決定する.

3.2.1 ドキュメントファイルをアップロード

Web 上で動く Ruby on Rails のシステムに対して, カテゴライズしたいドキュメントのファイルをアップロードする. ファイルのアップロード用のアップローダークラスを定義し, 詳しいオプションの設定を行う. アップロードされた画像ファイルのインスタンスをサービスクラスに渡し, OCR 等の処理を行う.

```
1    <div class="result">
2        <div class="result_content">この文書のカテゴリは
3        <%= @result_category %>です</div>
4        <div class="result_labels">参考データ：
5        <%= @category_labels %></div>
6    </div>
```

ソースコード 3.1: フロントエンドの ERB

3.2.2 OCR によってドキュメントの内容を取得

Google Vision API の OCR 機能を用いて, ドキュメントの内容を取得し, 単語ごとに分割する. API リクエストを送信し, 外部とやり取りをするためのサービスクラスを定義し, その中で処理を行う. detect_text クラスの初期化時に画像認識用のインスタンスを生成し, 画像のパスを取得する. 認識した画像に対しての処理を行い, OCR の結果を単語の配列として返す. API のレスポンスは, オブジェクトによって形成されているため, OCR の結果を保持した文字列を格納したプロパティを取得する. 最後に http のネットワークエラー処理を記述している.

```ruby
require "google/cloud/vision/v1"

class VisionOcrService
    def initialize(image_path)
        @image_path = image_path
        @vision = Google::Cloud::Vision::V1::
        ImageAnnotator::Client.new
    end

    def detect_text
        image_content = File.binread(@image_path)
        response = @vision.text_detection(image: {content:
        image_content})

        response.resources.flat_map do |res|
        res.text_annotations.map(&:description)
        end

    rescue StandardError => e
        Rails.logger.error "Vision API Error: #{e.message}"
        []
    end
end
```

ソースコード 3.2: Ruby による OCR の実装

3.2.3 それぞれの単語の定義の取得

WordNet のデータベースと連携し, 入力された単語の定義を取得する. アクションの実行前に,OCR によって取得した単語の配列を, インスタンス変数に格納する. 取得するラベルを格納するインスタンス変数を定義する. WordNet の Word テーブル内で, その単語に該当する定義を取得し, カテゴリが存在しない場合はカテゴリ無しとして取得する. 取得した Synset テーブルの一覧から, 該当するカテゴリを判別する. 上記の処理を、前の処理で取得した OCR 結果の単語の配列の全ての要素に対して行う.

```
1   class AnalyzeController < ApplicationController
2       before_action :set_words, only: [:analyze]
3
4       def analyze
5           @category_labels = Hash.new(0)
6
7           @words.each do |word|
8           analyze_word = Word.includes(:synsets).find_by(lemma:
9           word)
10          return showNoCategoryError if analyze_word.nil?
11
12          result_words = analyze_word.synsets.pluck(:name)
13          current_labels = label_category(result_words)
14
15          current_labels.each do |category, count|
16              @category_labels[category] += count
17          end
18          end
19          @result_category = get_category(@category_labels)
20      end
21
22      private
23      def set_words
24          @words = @ocr_response
25      end
26  end
```

ソースコード 3.3: ActiveRecord による WordNet との連携

3.2.4 単語ごとにカテゴライズ, ラベリング

取得した単語の定義から, 該当するカテゴリに対してラベル付けを行う. どの定義がどのカテゴリに該当するかは, Category テーブルに記述してある. 前の処理で取得した Synset の一覧に対して, 該当するカテゴリが見つかった際にラベルのカウントを増やす仕組みになっている. ラベリングを行ったハッシュを次の処理に渡し, 最終的なドキュメントのカテゴリを決定する.

```
1   def label_category(words)
2       words_set = words.to_set
3
4       categories = Category.all
5       words_with_synsets = Word.where(lemma:
6       categories.pluck(:value)).includes(:synsets)
7
8       synsets_by_category = words_with_synsets
9       .each_with_object({}) do |word, hash|
10      hash[word.lemma] = word.synsets.map(&:name)
11      end
12
13      categories.each_with_object({}) do |category,
14      category_labels|
15      synset_names = synsets_by_category[category.value] || []
16      label_count = synset_names.count { |name| words_set
17      .include?(name) }
18      category_labels[category.value] = label_count
19      end
20  end
```

ソースコード 3.4: カテゴリのラベリングメソッド

3.2.5 ラベリング結果から文章のカテゴリを決定

全ての単語をカテゴライズした後, 最もラベリングの数が多いカテゴリを文章のカテゴリとして決定する. 受け取ったハッシュに対して, ラベルの数が最も多いカテゴリを判別し, そのカテゴリをドキュメントの最終的なカテゴリとして決定する. なお, カテゴリが存在しない場合は, 現在はエラーをスローされるシステムにしている.

```
1    def get_category(labels)
2    max_label = labels.max_by { |_, value| value }
3    if max_label[1] == 0
4    showNoCategoryError
5    else
6    return max_label[0]
7    end
8    end
9
10   def showNoCategoryError
11   return 'no category'
12   end
```

ソースコード 3.5: 文書のカテゴリを決定するメソッド

第4章　実験 ・ 評価

4.1　様々なドキュメントでの実験

本研究の実験で用意するカテゴリは, 経理, 人事, 事務の 3 種類であり, 各カテゴリごとに 5 種類ずつの計 15 種類のドキュメントで行う.

1. 経理

   会社経理統制と経理検査

   工業経理規範

   戦時会社経理統制体制の展開

   法人税法の損金経理要件について

   陸軍経理組織の変遷と内部監査制度

2. 人事

   トヨタウェイと人事管理・労使管理

   公務員の人事異動と人材形成

   新・人事労務管理

   人事労務管理

   戦略的パートナーとしての日本の人事部

3. 事務

   ロボティックス・プロセス・オートメーションが事務職に及ぼす影響に関する一考察

   一般事務女性の職業生活意識に関する一考察

   女性事務職に得る派遣労働者の活用

   女性事務職のキャリア拡大と職場組織

   女性事務職の賃金と就業行動

4.2　精度の分析

4.2.1　タイトルごとの精度

タイトル	内容
会社経理統制と経理検査	一般企業の経理統制について (昔の字体が存在
工業経理規範	昔の経理の仕組みについて (字体は全て昔の書
戦時会社経理統制体制の展開	戦前の経理体制について (縦書き 2 列で書かれて
法人税法の損金経理要件について	法人税法という法律について
陸軍経理組織の変遷と内部監査制度	失敗
トヨタウェイと人事管理・労使管理	失敗
公務員の人事異動と人材形成	成功
新・人事労務管理	成功
人事労務管理	成功
戦略的パートナーとしての日本の人事部	成功
RPA が事務職に及ぼす影響に関する一考察	成功
一般事務女性の職業生活意識に関する一考察	成功
女性事務職に得る派遣労働者の活用	成功
女性事務職のキャリア拡大と職場組織	成功
女性事務職の賃金と就業行動	成功

表 4.1: タイトルごとの精度

4.2.2　カテゴリごとの精度

カテゴリ	正解率 (%)
経理	95
人事	80
事務	60

表 4.2: カテゴリごとの精度

第5章　考察

5.1　ドキュメントごとの結果の解釈

5.2　改善点

第6章　おわりに

6.1　今後の展望

6.2　研究のまとめ

参考文献

[1] D-Analyzer. RPA にできないこと,不得意な業務はなにか？ . https://tepss.com/danalyzer/column/rpa_can_not_do.html, 2019.

[2] Dave Thomas Noel Rappin. *Programming Ruby 3.3: The Pragmatic Programmers' Guide (Pragmatic Programmers; Facets of Ruby)*. Pragmatic Bookshelf, 5th edition, 2024.

[3] Sony. 機械学習における教師なし学習とは？ ディープラーニングとの関係と応用. https://dl.sony.com/ja/deeplearning/about/unsupervised_learning.html. 参照日 2024 年 12 月 12 日.

[4] まつもと ゆきひろ. オブジェクト指向スクリプト言語 Ruby. http://www.ruby-lang.org/ja/, 2023.

[5] 倉橋 和子. 分割・併合機能を有する k-means アルゴリズムによるクラスタリング. 2007.

[6] 大西 淑雅. 文字認識 API を用いた講義アーカイブシステムの設計. 2018.

[7] 竹内 孔一. 日本語 wordnet における語義・概念の分散表現獲得. 2019.

[8] 佐々木 康浩. RPA (Robotics Process Automation) の可能性. 2017. 参照日 2024 年 12 月 12 日.

[9] 東埜 淳哉. 階層的クラスタリングを用いたクラスタ数の自動推定に関する検討. 2022.

[10] 高橋 明日香. フレームワークを用いた Web アプリケーション開発における変更容易性の評価. 2013.