書類整理の自動化

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

本章では，本研究を始めるにあたっての動機，および本論文の構成を示す.

1.1　研究背景

日本企業の RPA(Robotics Process Automation [7]) 導入率は全体で 38%, 中堅・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, 大企業と中堅 ・ 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ｚ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.

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.

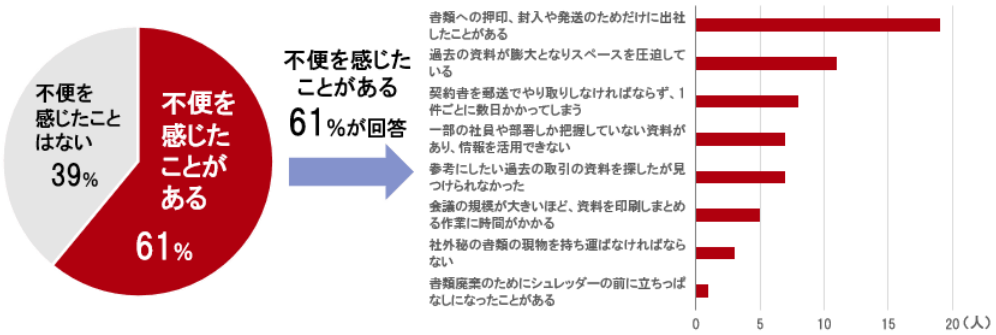


図 1.2: 紙媒体中心業務に関するアンケート

第2章　関連研究

本章では, クラスタリングの手法として有名な手法や課題, 使用する基礎技術を示す.

2.1　既存手法

2.1.1　クラスタリングの既存手法

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. [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. [9]

2.1.2　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 [4] 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.

2.2 基礎技術

2.2.1 Ruby

Ruby [8] 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．

2.2.2 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. [5]

2.2.3 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. [3]

2.2.4 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. [6]

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 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　提案する手法 ・ アプローチ

本研究での手法は、以下のステップで構成されます.

1. ドキュメントファイルをアップロード
2. **OCR** によってドキュメントの内容を取得
3. それぞれの単語の定義の取得
4. 定義ごとにカテゴライズ、ラベリング
5. ラベリング結果から文章のカテゴリを決定

3.1.1　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.

```
1   <div class="result">
2       <div class="result_content">この文書のカテゴリは
            <%= @result_category %>です</div>
3       <div class="result_labels">参考データ：
            <%= @category_labels %></div>
4   </div>
5
6   <style>
7       .result {
8           background-color: grey;
9           text-align: center;
10          display: flex;
11          flex-direction: column;
12          justify-content: center;
13          height: 100vh;
14      }
15      .result_content {
16          font-size: 30px;
17          font-weight: bold;
18      }
19
20      .result_labels {
21          margin-top: 10px;
22          font-size: 20px;
23      }
24  </style>
```

ソースコード 3.1: フロントエンドの ERB

3.1.2 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.

```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

3.1.3 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.

```ruby
class AnalyzeController < ApplicationController
    before_action :set_words, only: [:analyze]

    def analyze
        @category_labels = Hash.new(0)

        @words.each do |word|
        analyze_word = Word.includes(:synsets).find_by(lemma:
            word)
        return showNoCategoryError if analyze_word.nil?

        result_words = analyze_word.synsets.pluck(:name)
        current_labels = label_category(result_words)

        current_labels.each do |category, count|
            @category_labels[category] += count
        end
        end
        @result_category = get_category(@category_labels)
    end

    private
    def set_words
        @words = @ocr_response
    end
end
```

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

3.1.4 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.

```ruby
def label_category(words)
    words_set = words.to_set

    categories = Category.all
    words_with_synsets = Word.where(lemma: categories.pluck(:value)).includes(:synsets)

    synsets_by_category = words_with_synsets.each_with_object({}) do |word, hash|
    hash[word.lemma] = word.synsets.map(&:name)
    end

    categories.each_with_object({}) do |category, category_labels|
    synset_names = synsets_by_category[category.value] || []
    label_count = synset_names.count { |name| words_set.include?(name) }
    category_labels[category.value] = label_count
    end
end
```

ソースコード 3.4: カテゴリのラベリングメソッド

3.1.5 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.

```
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
```

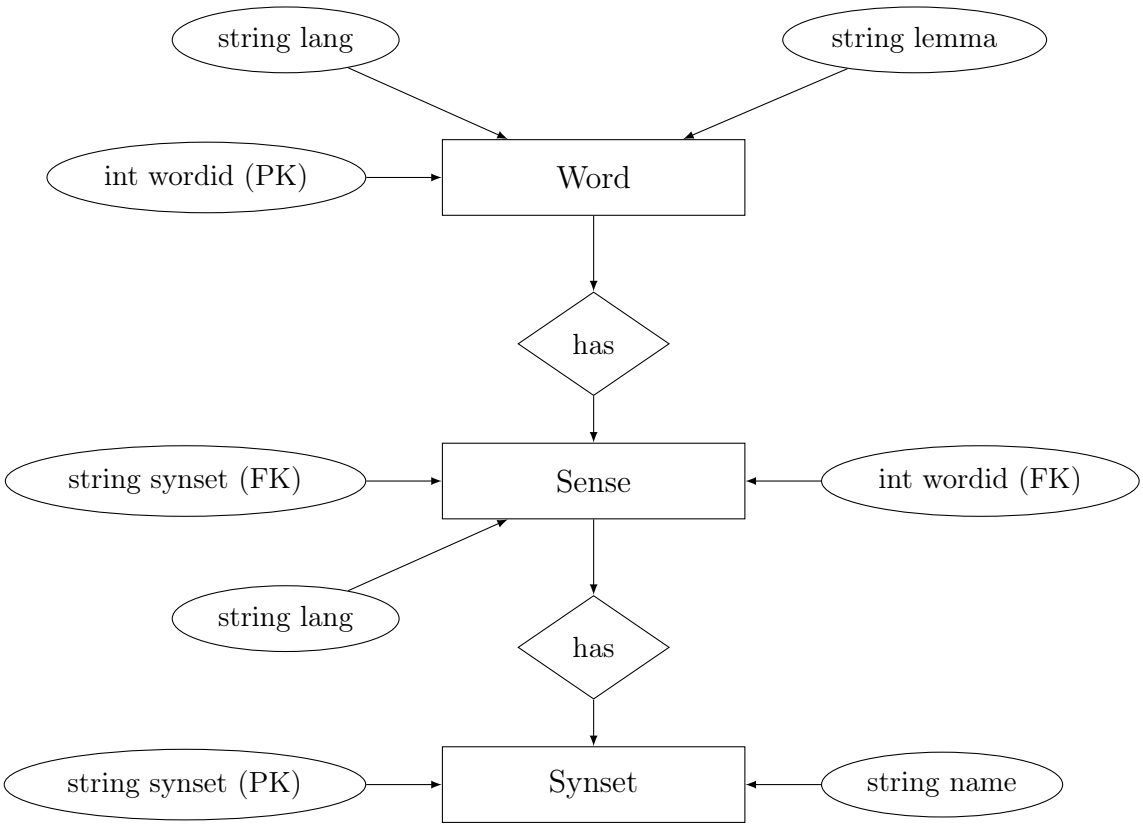
ソースコード 3.5: 文書のカテゴリを決定するメソッド

3.2 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.

3.2.1 テーブル構造

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



WordNet テーブルの ER 図

3.2.2 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.

第4章　実験・評価

4.1　様々なドキュメントでの実験

4.2　精度の分析

4.3　既存手法との比較

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] Kazuko kurahashi. 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.

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

[4] 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.

[5] Takahashi asuka. 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.

[6] Koichi Takeuchi. 日本語 wordnet における語義・概念の分散表現獲得. 2019.

[7] yasuhiro sasaki. RPA(Robotics Process Automation) の可能性. *2017*年春季全国研究発表大会, 2017.

[8] 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.

[9] 東京工業大学情報理工学院. 階層的クラスタリング. https://chokkan.github.io/mlnote/unsupervised/02hac.html, 2021.