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

学生番号：15820094

嘉松 一汰

目 次

第 1 章　はじめに **1**
1.1　研究背景 1
1.2　研究目的 2
1.3　論文の構成 3

第 2 章　関連技術 **4**
2.1　基礎技術 4
2.1.1　Ruby 4
2.1.2　Ruby on Rails 4
2.1.3　OCR (Optional Character Recognition) 4
2.1.4　Google Cloud Vision API 5
2.1.5　WordNet 5
2.2　クラスタリングの既存手法 5
2.2.1　非階層的クラスタリング 5
2.2.2　階層的クラスタリング 5

第 3 章　関連研究 **6**
3.1　カテゴライズに関する研究 6
3.1.1　機械学習を用いたカテゴライズ 6
3.1.2　ドキュメントのカテゴライズ 6
3.2　先行研究との比較 7
3.2.1　先行研究との類似点 7
3.2.2　先行研究との差異 7

第 4 章　提案手法 **8**
4.1　システム設計 8
4.1.1　データベース設計 8
4.1.2　モデル設計 9
4.2　提案する手法 ・ アプローチ 10
4.2.1　ドキュメントファイルをアップロード 10
4.2.2　OCR によってドキュメントの内容を取得 11
4.2.3　それぞれの単語の定義の取得 12

4.2.4　単語ごとにカテゴライズ，ラベリング 13
4.2.5　ラベリング結果から文章のカテゴリを決定 13

第 5 章　実験・評価 **15**
5.1　精度の分析 15
5.1.1　タイトルごとの精度 15
5.1.2　カテゴリごとの精度 16

第 6 章　考察 **17**
6.1　ドキュメントごとの結果の解釈 17
6.2　改善点 17

第 7 章　おわりに **19**
7.1　研究のまとめ 19
7.2　今後の展望 19

謝辞 **20**

参考文献 **21**

付録 実験で使用した資料 **22**

第1章　はじめに

1.1　研究背景

日本企業の RPA (Robotics Process Automation [1]) 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，アナログの世界で実施さ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処理することは，現代では非常に困難なことである．具体的には，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する必要があるからである [2]．2 つ目の原因は，紙媒体の業務を実施している企業の 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．また， RPA の定義についても，明確には定まっていない部分が多い．自動化という概念自体に対して RPA という言葉を適用するとしたら，工場で食品の生産や梱包をするロボットも，我々の家で活躍するお掃除ロボットも， PC 上のボタン一つで複数の処理をするシステムも全て RPA と呼ぶことができる．そのため，本研究では，RPA という言葉の意味を広義的に捉え， RPA の非専門領域で実施される処理を，他の様々な技術を用いて克服することを目標とする．より詳細な技術については第 2 章で述べるが， OCR や 字句解析の技術を使用し，紙媒体に対する処理を不自由なく自動化し， Ruby on Rails を使用したシステムとしての運用をして，我々の周りに多くある Web アプリケーションと同様に利用できるようにすることで，IT に関する知識が乏しい企業でも，安全かつ快適に自動化の恩恵を受けることを目指す.

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，紙媒体の業務を実施している企業を中心に，RPA を使用して紙媒体の処理を自動で実施するシステムを作成することである．具体的には，紙媒体の処理を OCR (Optical Character Recognition) 技術を用いてデジタルデータに変換し，さらに文字認識結果を用いた自動分類・ラベリングによって，手作業の削減と業務効率の向上を目指す．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る．多くの企業がデジタル化に対する適応を進めている中，依然として紙媒体を中心とした業務フローに頼らざるを得ない状況がある．これにより，文書の管理や検索に時間がかかるだけでなく，人的ミスや紛失のリスクも存在する．また，既存の RPA ソリューションでは紙媒体の取り扱いが難しく，専用の高価な機器が必要となる場合がある．本研究では，これらの課題を克服し，誰でも簡単に使用できるシステムを設計・実装する．紙ベースの情報管理を効率化し，日常業務で活用できるようにする．さらに，システムを Web アプリケーションとして提供し，スマートフォンやタブレットからも簡単にアクセス可能な形にすることで，利便性向上を目指す．このような RPA ソリューションを普及させることにより，デジタル化が遅れている分野でも手軽に導入できる環境を提供し，業務の自動化を促進することを目指す．

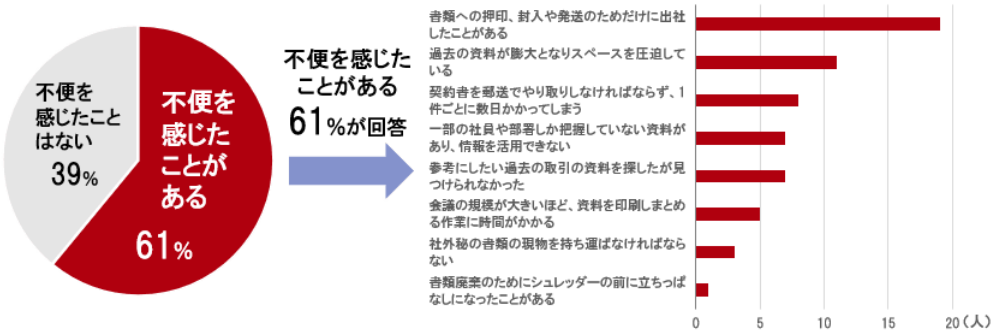

図 1.2: 紙媒体中心業務に関するアンケート

1.3 論文の構成

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た．第 2 章では本研究で使用した基礎技術，関連技術を述べる．第 3 章では，先行研究について説明し，本研究との差異を述べる．本研究での実装手法を第 4 章で提案し，第 5 章で実験と評価，第 6 章で結果の考察を述べ，第 7 章でに本研究の総括をする．

第2章　関連技術

2.1　基礎技術

2.1.1　Ruby

Ruby [3] [4] 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 [5]．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　OCR (Optional Character Recognition)

印刷された文字や手書きの文字などをカメラやスキャナといった光学的な手段でデータとして取り込み，それを解読（文字認識）することによって一度印刷されてしまった文字をパソコンなどのコンピューターが利用できる文字（テキスト）データに変換する技術．データ入力作業の手間を大幅に削減し 2 重入力や人的ミスの削減などを目的とした利用はビジネス用途にも広く浸透しており，流通・製造・医療・小売などあらゆる業界で本来は人が読むために印刷された文字をコンピューターに取り込みたいといった要望が根強くあり，バーコードや 2 次元コードが普及した現在でも需要はむしろ高まる傾向にあります．

2.1.4 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 処理をして，単語ごとに分類するために使用する．

2.1.5 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 (Synset)，語彙テーブル (Word) とそのリレーションテーブル (Sense) によって形成されている [7]．本研究では，単語を該当するカテゴリに分け，ラベリングをするために使用する．

2.2 クラスタリングの既存手法

本研究では，あくまでカテゴライズという目的でシステムを構築しているため，クラスタリング手法とは少し異なる点があるが，アルゴリズム等の観点で参考にしたため，関連技術として以下に示す．

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処理さ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 [8]．

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処理する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く，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]．

第3章　関連研究

3.1　カテゴライズに関する研究

3.1.1　機械学習を用いたカテゴライズ

花房 良ら [10] は，プログラミング読解中の視線軌道を機械学習によりカテゴライズすることを試みた．従来，プログラミング技能の評価は主観的な評価や定性的な分析が中心であった．しかし，視線追跡データを用い，教師あり学習を用いた学習モデルの 1 つである SVM (Support Vector Machine) を活用することにより，技能の異なる学習者（得意群・普通群・不得意群）の視線軌道パターンの定量的に分類した．大学 4 年生 24 名を対象とし，C 言語のプログラムを提示しながら視線の動きを計測した．ヒートマップデータを簡略化し，視線パターンの特徴を抽出した．

任意のデータを定量的に分類し，精度を分析している点は，本研究と共通している．主観的評価に依存せず客観的な分析を実現している点は，機械学習によるカテゴライズの利点といえるだろう．一方で，低難易度の問題では分類精度が低下したという結果から，解析結果の信頼性が，使用するデータセットに依存してしまう点は，カテゴライズの課題点だといえる．

3.1.2　ドキュメントのカテゴライズ

松田 勝志ら [11] は，Web ページ (HTML 文書) を，HTML タグと，その要素の情報によってカテゴライズすることで，WWW 検索システムの拡張を試みた．従来の検索の仕組みは，検索キーワードと全てのページを照合し，一致率の高い順番で表示する．しかし，あらかじめ，Web ページをいくつかのカテゴリに分類しておくことで，検索速度や，内容一致度の向上が見込まれる．

処理の具体的な流れを示す．カテゴライズの基準となる特徴記述ファイルを用意する．ファイル内には，タグと要素の組み合わせごとに，どのカテゴリに分類されるかの詳細な条件が記述されている．読み込んだ HTML ファイルと特徴記述ファイルを比較し，文書のカテゴリを決定する．

文章に対して内容を解析し，カテゴライズするという点が本研究と共通している．特に解析方法について，Web ページ全体を複数の要素に分割し，各々に対して処理をした後に，文書全体のカテゴリを判別するという流れは，本研究でも取り入れている．

3.2 先行研究との比較

3.2.1 先行研究との類似点

先行研究 [10] [11] と同様に，特定の媒体に対してカテゴライズを実施しており，主観的な判断を排除しデータに基づく定量的な結果の分析を重視している．特に先行研究 [11] では，文書を要素ごとに分割し、それぞれのカテゴリを判別する手法を採用しているのに対し、本研究でも文書を単語レベルに分割し、各単語の定義を解析することで文書全体のカテゴリを決定するアプローチを採用している。

3.2.2 先行研究との差異

一方で，これらの研究との大きな違いは，システムのコストパフォーマンスと拡張性である．先行研究 [10] では，機械学習モデルを使用しているが，このような高い精度を担保するモデルを 1 つの処理ごとに使用すると，時間的，金銭的にも高コストになってしまう．本研究では，そのような機械学習モデルは使用しておらず．少ないリソースで構築可能なシステムになっている．さらに，拡張性に関しても，機械学習モデルを使用している場合はそのモデルにある程度依存した結果しか得られないが，本研究ではカテゴライズの条件を自由にカスタマイズできるため，各々の企業に最適なシステムを追求することができる．

また，先行研究 [11] では，自身でカテゴライズの基準となるファイルを作成する必要があり．カテゴリが増えるたびに判別条件を考える必要がある．本研究では，単語の定義からカテゴライズするため，カテゴリに該当する単語を容易に取得することができるため，システムの構築や拡張段階において，コストを低く抑えることができる．

第4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ライズするかを具体的に示す．

4.1　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．

4.1.1　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．図 4.1 に，これらを表す ER 図を示す．

Word テーブル　Wordnet データベースの単語が格納されており，主キーである整数型の wordid カラム，言語を表す文字列型の lang カラム，単語名を表す文字列型の lemma カラムによって形成されている．

Synset テーブル　WordNet データベースに含まれる単語の定義が格納されており，主キーである文字列型の synset カラム 定義名を表す name カラムによって形成されている．

Sense テーブル　前述した 2 つのテーブルの中間テーブルであり，Word テーブルとの外部キーである wordid カラム，Synset テーブルとの外部キーである synset カラム，言語を表す文字列型の lang カラムによって形成されている．

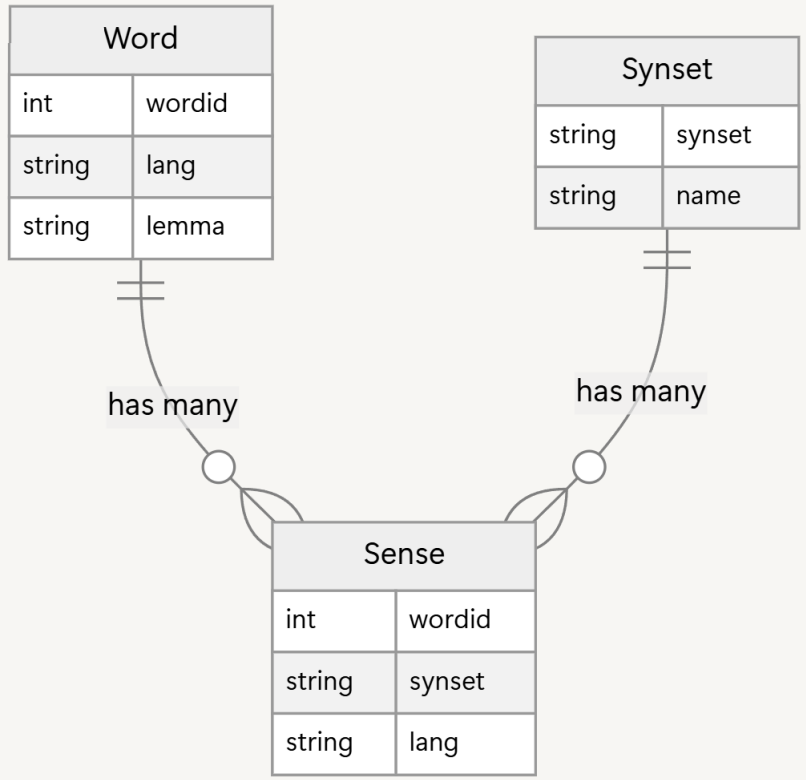

図 4.1: データベースの ER 図

4.1.2 モデル設計

Ruby on Rails にはモデルという概念があり，データベースの内容をモデルとして定義することで， CRUD 処理を始めとしたバックエンドでのデータベースのやり取りを効率的にすることができる．本研究では第 4.1.1 節で示したテーブルをモデルとして定義し，整合性等のデータベースレベルの具体的なプロパティを設定する．

Word モデル　単語に関する情報が格納されている．Sense モデルと一対多の関係で，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するスコープを持つ．

Synset モデル　単語の定義に関する情報が格納されている．主キーは文字列型の synset カラムである．Sense モデルと一対多の関係で，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するスコープを持つ．

Sense モデル　Sense モデルは，上記の 2 つのモデルそれぞれと一対多のリレーションを持っている．synset カラムと wordid カラムが格納されており，それらを外部キーとして使用している．カラムと 2 つのモデルの双方向に紐づける役割を担う．単語と定義間で実行される処理は，Sense モデルを経由する．

4.2　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 OCR 処理をして，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，Google Vision API を本研究で使用する．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することができるため，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する．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．なお，各々のカテゴリに関しては，WordNet の Synset モデルにカテゴリ名を入力し，単語群を取得する方法で用意している．

4.2.1　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する．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 等の処理をする．上記の処理の流れをソースコード 4.1 に示す．

```
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 4.1: フロントエンドの ERB

4.2.2 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．本研究では，まず OCR を実行するために Google Vision API を活用し，外部サービスと連携するためのサービスクラスを設計する．このサービスクラス内で，API リクエストの送信処理やレスポンスの解析処理を行う．

まず VisionOcrService クラスのインスタンスを初期化する際に，画像認識のための ImageAnnotator::Client インスタンスを生成し，OCR を行う画像のパスを取得する．次に，指定された画像ファイルの内容をバイナリ形式で読み込み，Vision API にリクエストを送信する．API のレスポンスは，テキスト認識結果を保持するオブジェクトとして返される．このオブジェクト内から，OCR 結果として取得した文字列データをプロパティから抽出し，単語の配列として整形する．処理の最終段階では，API 通信時のエラー処理として，ネットワークエラーや API 側の制限に伴う例外処理を行い，適切なエラーメッセージをログに記録することで，システムの信頼性を向上させる．詳細な処理の流れをソースコード 4.2 に示す．

```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 4.2: Ruby による OCR の実装

4.2.3　それぞれの単語の定義の取得

OCR によって取得した単語に対し，WordNet のデータベースを用いて定義を取得する．まず OCR の結果を配列形式でインスタンス変数に格納し，その後，分析プロセスを開始する．

具体的には，まず単語リストに対して，WordNet の Word テーブルから該当するエントリを検索し，関連する定義情報を取得する．定義情報が見つからない場合は，"カテゴリ無し" として処理する．これらの処理は，OCR 結果のすべての単語に対して適用される．最終的に，Synset テーブルの一覧をもとに，文書のカテゴリを判定するためのデータが整えられる．詳細な処理の流れをソースコード 4.3 に示す．

```
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 4.3: ActiveRecord による WordNet との連携

4.2.4　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を基に，文書を適切なカテゴリに分類するためのラベリングを行う．カテゴライズ処理では，WordNet から取得した Synset 情報をもとに，どの単語がどのカテゴリに属するかを決定する．システムは，各 Synset を解析し，それがどのカテゴリに該当するかを判別する．この過程では，単語リストとカテゴリのマッピングをハッシュ形式で保持し，各カテゴリのカウントを増加させることで，どのカテゴリが最も関連性が高いかを測定する．

具体的には，単語を一意の集合として扱い，カテゴリごとの Synset 情報を取得し，対応するカテゴリラベルの数をカウントする．これにより，文書内の単語がどのカテゴリに属しているかを判別し，最終的な文書のカテゴリ決定へと繋げる．詳細な処理の流れをソースコード 4.4 に示す．

```
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 4.4: カテゴリのラベリングメソッド

4.2.5　ラベリング結果から文章のカテゴリを決定

ラベリングされた結果をもとに文書の最終的なカテゴリを決定する．すべての単語に対してカテゴリが割り当てられた後，カテゴリごとのラベルカウントを集計し，最も多く出現したカテゴリを文書の主要カテゴリとして選定する．

まず受け取ったラベルデータのハッシュを解析し，カウント数が最大のカテゴリを特定する．この処理の際，カテゴリが全く見つからなかった場合には，エラー

をスローし，適切なメッセージを表示することで，未分類の文書に対する処理を明確化する．最終的に，識別されたカテゴリは，文書全体のカテゴリとして出力される．詳細な処理の流れをソースコード 4.5 に示す．

```
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 4.5: 文書のカテゴリを決定するメソッド

第5章　実験・評価

5.1　精度の分析

5.1.1　タイトルごとの精度

さまざまなフォーマットや内容の文書を対象にOCR処理を実施し，タイトルごとの処理精度を評価した．その結果を表5.1に示す．

タイトル	結果	備考
会社経理統制と経理検査	失敗	昔の字体が存在する
工業経理規範	失敗	字体は全て昔の書き方
戦時会社経理統制体制の展開	成功	縦書き2列構成
法人税法の損金経理要件について	成功	減価償却等の専門的な情報
陸軍経理組織の変遷と内部監査制度	成功	
トヨタウェイと人事管理・労使管理	成功	縦書き2列構成
公務員の人事異動と人材形成	成功	図や英語が混在
新・人事労務管理	成功	フォントがゴシック体の太文字
人事労務管理	成功	縦書きと横書きが混在
戦略的パートナーとしての日本の人事部	成功	図や見出しが英語表記
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	見出しのみ英語表記
女性事務職に得る派遣労働者の活用	成功	縦書き2列構成のゴシック体
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 5.1: タイトルごとの精度

表5.1に示したように，処理精度にはタイトルごとにばらつきが見られる．特に，「会社経理統制と経理検査」や「工業経理規範」のように，旧字体を含む文書では処理が失敗する傾向が確認された．一方で，縦書きや横書きが混在する文書，英語表記が一部含まれる文書，ゴシック体や太文字を用いた文書などにおいては，

ほとんど問題なく処理が成功している．これは，現代の標準的なフォーマットに近い文書に対しては，OCR の処理が適切に機能することを示している．また，縦書き 2 列構成や図表が含まれる場合も成功率は高く，特に人事関連のタイトルにおいては安定した処理精度が見られた．これらの結果から，フォーマットや字体の違いが OCR の精度に与える影響が明確になった．

5.1.2 カテゴリごとの精度

続いて，文書をカテゴリに分類し，それぞれのカテゴリごとに処理精度を評価した結果を表 5.2 に示す．

カテゴリ	精度 (%)
経理	60
人事	100
事務	100

表 5.2: カテゴリごとの精度

表 5.2 に示したように，経理カテゴリにおける精度が他のカテゴリと比べて著しく低いことが確認された．この主な要因としては，経理カテゴリの文書に旧字体や特殊な専門用語が多く含まれている点が挙げられる．一方で，人事や事務カテゴリにおいては，フォーマットが比較的一般的であるため，100%の精度を達成している．さらに，精度が高いカテゴリにおいても，処理内容には軽微な誤解釈が存在する場合があるが，全体としてカテゴリごとの分類精度に影響を与えるような問題は確認されなかった．これにより，OCR 処理のカテゴリごとの安定性について評価することができる．以上の結果から，タイトルごとの処理精度とカテゴリごとの精度において，それぞれの特性が確認された．本研究では，特定の条件下における処理の成功率を明らかにしたが，さらなる精度向上を目指すためには，旧字体や特殊なフォーマットへの対応が課題として残る．これらの詳細な考察については次章で述べる．

第6章　考察

6.1　ドキュメントごとの結果の解釈

第 5 で本研究で実施した実験を紹介したが，本章ではその結果から，本研究で提示したシステムの優位点及び将来性や有用性について述べる．まず，実験により明らかとなった本システムの優位点について述べる．それは，ドキュメントの形式に関わらず同様の処理結果を得ることができる点である．文書には，縦書きや横書き及びそれらの複合など，内容の形式はジャンルによって多岐にわたる．それら全てに対応出来なければ，本システムの優位性は著しく低下してしまう．しかし，さまざまな形式の文書を実験では用意したが，特に形式の差による精度の変化は見受けられなかった．そのため，本システムにおいては，文書をアップロードする前に形式を変換する必要がなく，その点ではストレスなくシステムを運用することができると考えられる．本システムの将来性については，発展途上であるという結論になってしまうが，将来的にカテゴリが増え，解析の幅が広がることや，WordNet データベースがアップデートされ，扱うことのできる語彙が増えていくことを考えると，IT 分野の発展に伴い，相対的にニーズが増えていくと考えられる．本システムの有用性については，第 1 で大まかに述べたが，紙媒体中心の業務に不便を感じている人が現代に多くいるため，そのような状況を改善するという観点では，要件を満たすことができているといえるだろう．

6.2　改善点

本システムの欠点について挙げられることは，旧字体など，現在使用されていない字体を用いた文書に関しては，精度が落ちてしまう点である．具体的には，OCR 処理では不自由なく単語ごとに分けることができているが，WordNet データベースにその単語が存在しないため，単語のカテゴリが判別できないことが原因として挙げられる．また，機械学習を活用して未知の字体を動的に解析する技術を導入することも検討されるべきである．しかし，あくまで本システムは現代の企業を対象としており，旧字体が記入された文書を取り扱うことは極めて少ないため，エンドユーザー視点では大きな欠点ではないといえる．この欠点を改善するため

には，現状使用している WordNet データベースの他に，漢字字体規範史データセットのような旧字体を扱っているデータベースを使用し，判別可能な語彙を拡張することが必要である．これらを中心とした複数のデータベースを 1 つの MVC アーキテクチャに統合することができれば，本システムの有用性や汎用性の向上につながると考えられる．また，現状のシステムでは，日本語と英語を主な対象としており，WordNet データベースを利用してカテゴリの解析をしている．しかし，グローバル化が進む現代社会では，多言語対応が求められる場面が増えている．特に，アジア地域で広く使用されている中国語，韓国語に対応することで，システムの利用範囲をさらに拡大することが可能である．これを実現するためには，各言語に特化した語彙データベースを統合し，多言語間でのカテゴリ解析を可能にするアルゴリズムの開発が必要である．

第7章　おわりに

7.1　研究のまとめ

本研究で開発したシステムは，現在の業務フローにおける課題を解決し得る有効なツールであると結論付けられる．また，さらなる改良を通じて，専門分野や多言語対応，クラウドサービス化といった新たな領域への応用が期待される．現代では，今後も IT 分野の著しい成長に遅れをとった企業を中心に，さまざまな業務的ニーズが発生すると考えられる．本研究では紙媒体中心業務をさらに細分化した，ドキュメントのカテゴライズ分野に着目してシステムの作成をしたが，本システムのようなソリューションが生み出されることで，IT 社会の課題が解決されることを願っている．

7.2　今後の展望

本システムを現実的な業務フローに組み込むには，ユーザーインターフェース（UI）の最適化も重要である．現状ではシステムのコア機能に重点を置いているが，操作の直感性やユーザーエクスペリエンスの向上が求められる．例えば，解析結果を可視化するダッシュボードや，フィードバック機能を備えたインターフェースを実装することで，利用者の利便性を高めることが可能である．処理の精度について，経理，人事，処務といったカテゴリにおいて，高い精度での解析が可能であることが確認された．本システムはさらに分野ごとに特化した最適化を施すことで，専門分野への応用を推進できる，例えば医療分野や法律分野では特有の専門用語や文書構造が存在するため，分野別にカスタマイズされた解析アルゴリズムが必要である．また，本システムをクラウドベースで提供することにより，複数のユーザーが同時に利用できる環境を構築し，スケーラビリティを確保することができる．また，API を公開することで，他のアプリケーションやサービスとの統合を可能にし，システムの拡張性をさらに高めることも有効である．

謝辞

本研究の遂行にあたり，多大なるご指導とご助言を賜りました指導教員の Martin J. Dürst 教授に心より感謝申し上げます．研究の方向性について貴重なご意見をいただき，また研究内容の精査においても適切なアドバイスをいただきましたことに深く感謝いたします．また，研究活動において協力を惜しまなかった研究室の皆様に感謝申し上げます．議論を通じて新たな視点を得ることができ，研究の深化に大いに寄与していただきました．さらに，本論文の作成にあたり添削をしてくださった石井幹大助手に心より感謝申し上げます．皆様の支えがなければ，本システムの開発は実現し得なかったことを強く認識しております．本研究が，多くの人々のご支援とご協力によって成り立っていることを改めて認識し，深く感謝の意を表します．

参考文献

[1] 佐々木康浩. RPA (Robotic Process Automation) の可能性. 経営情報学会全国研究発表大会要旨集, Vol. 2017s, pp. 201–204, 2017.

[2] 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. 参照日 2024 年 12 月 12 日.

[3] 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. 参照日 2024 年 12 月 12 日.

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

[5] 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. 第 2013s 巻, pp. 1–7, 2013.

[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. 第 2018s 巻, pp. 1–6, 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. 第 18 巻, pp. 245–246, 2019.

[8] 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. 第 2007s 巻, pp. 1–36, 2007.

[9] 東埜淳哉, 幸裕. 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. 日本知能情報ファジィ学会 ファジィ システム シンポジウム 講演論文集, Vol. 38, pp. 679–684, 2022.

[10] 花房亮, 山岸秀一, 松本慎平, 加島智子. 機械学習処理に基づいたプログラミング読解中の視線軌道の自動分類. 人工知能学会全国大会論文集, Vol. JSAI2015, pp. 3N32–3N32, 2015.

[11] 勝志松田, 俊一福島. 文書タイプ分類による問題解決向き www 検索システムの開発と評価. Technical Report 20(2007-FI-053), NEC ヒューマンメディア研究所, NEC ヒューマンメディア研究所, mar 2007.

付録: 実験で使用した資料

経理

- 久保田秀樹, 「会社経理統制令と経理検査」, 山内隆教授退官記念論文集, 第 329 号, pp. 169-172, 1934 年
- 大野巖, 「工業経理規範」, 昭和 13 年 9 月, 化学機械, 第 2 巻 第 3 号, 1938 年
- 柴田雅,「戦時会社経理統制体制の展開」, The Socio-Economic History Society, 1937 年
- 前原真一, 『法人税法の損金経理要件について』, 税務大学校, 研究部教授
- 建部宏明, 「陸軍経理組織の変遷と内部監査制度Ⅱ」, 明治大学経理研究所, 2009 年

人事

- 猿田正機, 「トヨタウェイと人事管理・労使関係」, 税務経理協会, 2007 年
- 圓生和之, 「公務員の人事異動と人材形成―大卒ホワイトカラーの公民比較からの分析」, 日本労働研究雑誌第 759 号, pp. 47-53, 2023 年
- 津田眞澂, 「新・人事労務管理」, 有斐閣, 1995 年
- 津田眞澂編著, 「人事労務管理」, ミネルヴァ書房，1993 年
- 平野光俊, 「戦略的パートナーとしての日本の人事部―その役割の本質と課題―」, 神戸大学大学院経営学研究科，2010 年

処務

- 所正文,「一般事務職女性の職業生活意識に関する一考察」, 経営行動科学第 3 巻第 1 号, 1988 年
- 古武真美,「女性事務職における派遣労働者の活用」, 近畿大学短期大学論集第 44 巻第 1 号, pp. 11-20, 2011 年
- 浅海典子,「女性事務職のキャリア拡大と職場組織」, 日本経済評論社, 2006 年
- 寺村絵里子,「女性事務職の賃金と就業行動」, 国際短期大学人口学研究第 48 号, 2012 年
- 日高義浩,「RPA が事務職に及ぼす影響に関する一考察」, 鹿児島経済論集第 63 巻第 1 号, 2022 年