2021年度 卒業論文 2022/02/12 12:09

OCRを用いた賞味期限管理システムの提案

大阪産業大学 デザイン工学部 情報システム学科
情報教育システム研究室

18H047 佐川祥吾

OCR を用いた賞味期限管理システムの提案

18H047 佐川祥吾

1　はじめに

私たちは日常生活において必ず冷蔵庫などを使い、食品などを保存している。日々の食事の際に食品を取り出し調理、または電子レンジなどで温め食べている。しかし時には外出先での食事をすることもある。その際に以前に購入しておいたものを冷蔵庫に入れたまま賞味期限が過ぎて破棄せざるおえない時がある。そこで賞味期限を撮影することで簡単に食品の期限管理ができるものが欲しいと感じ、また現時点で存在するアプリケーションは賞味期限は入力して設定するため賞味期限を自動で設定できるシステムの作成をした。

2　目的

本研究の目的は、食品の賞味期限の管理をシステム化し家庭内の食品ロスを防ぐシステムを開発することである。日常生活において食品は日々購入し食べる日まで保存をする。しかし時には外食をしたりその日の気分で食材を変更し食べないまま賞味期限を迎えてしまうことがある。

農林水産省による食品ロスのアンケート [1] の結果では、”消費・賞味期限内に食べられなかった”,”購入後、冷蔵庫や保管場所に入れたまま存在を忘れてしまった”という原因が主にあげられる。

また現時点で存在する賞味期限管理アプリケーションは、賞味期限を手打ちで入力するものが多い。その点に注目し開発を行なった。

3　システム

本研究では、賞味期限を撮影し Raspberry Pi に送信することで自動的に日付を読み取り賞味期限日に近づくと通知が届くシステムである。Raspberry Pi では、OCR で賞味期限を読み取るプログラム、Gmail を送信するプログラム、この二つのプログラムを定期的に動かすプログラムの 3 つで構成されている。

スマートフォンで賞味期限を撮影し、Raspberry Pi に送信する。送信された画像から OCR で年月日を取得する。取得された年月日から 3 日減算し、JSON ファイルを書き出す。書き出された JSON ファイルは毎日午前 9:00 に読み取られ当日の場合メールを作成し、画像を添付して送信される。

4　検証

食品には様々な賞味期限表記があるため OCR を使用し、どのような記載が読み取れるのか OCR の検証を行なった。

5　結果と考察

本研究の結果、賞味期限を読み取ることで自動的に期限前日に通知が送られ消費することができた。しかし賞味期限がはっきりと記載されているもの以外は読み取ることができないとわかった。特にドット字体の場合は読み取ることができなかった。また近くに別の数字やアルファベットが記載されている場合は日付として読み取ることができなかった。

6　まとめ

本研究では食品の賞味期限を読み取ることで自動的に通知が来るシステムを開発した。その結果、冷蔵庫の奥にある食材の賞味期限が近づいていることが把握することができた。今後の課題として、OCR の精度向上やスマートフォンで写真を撮り PC から Raspberry Pi へ送るという手間をなくす、野菜や魚などの賞味期限の記載がないものは管理できないという問題点がある。

参考文献

[1] 農林水産省, 食育に関する意識調査報告書 https://www.maff.go.jp/j/syokuiku/ishiki/h29/zuhyou/z7-6.html

目次

1　はじめに

私たちは日常生活において必ず冷蔵庫を使い、食品などを保存している。日々の食事で冷蔵庫から食品を取り出し調理、または電子レンジなどで温め食べている。しかし時には外出先での食事をすることもある。その際に以前に購入しておいたものを冷蔵庫に入れたまま賞味期限が過ぎて破棄せざるおえない時がある。また、既存のアプリケーションで賞味期限は手打ちで入力するもが多い。

第 2 章では本研究の目的について、第 3 章では食品ロスについて、第 4 章ではシステムの概要、第 5 章には検証について、第 6 章には検証の結果のまとめについて、第 7 章には研究の成果とともに今後の課題について述べる。

2　目的

本研究の目的は、食品の賞味期限の管理をシステム化し家庭内の食品ロスを防ぐシステムを開発することである。日常生活において食品は日々購入し食べる日まで保存をする。しかし時には外食をしたりその日の気分で食材を変更し食べないまま賞味期限を迎えてしまうことがある。

また現時点で存在する賞味期限管理アプリケーションは、賞味期限を手打ちで入力するものが多い。その点に注目し開発を行なった。

3　食品ロス

現在日本での食品ロスの量は年間で約 600 万トンも存在する。その中でも家庭での料理の作り過ぎによる破棄や賞味期限切れによる破棄、買い過ぎによる破棄など家庭内食品ロスが年間で約 276 万トンもある。そこで日本では SDGs [1] によって 2000 年度比で 2030 年度までに半減させることを目標としている。過去年度の食品ロス量の推移と削減目標を**図 1** に示す。

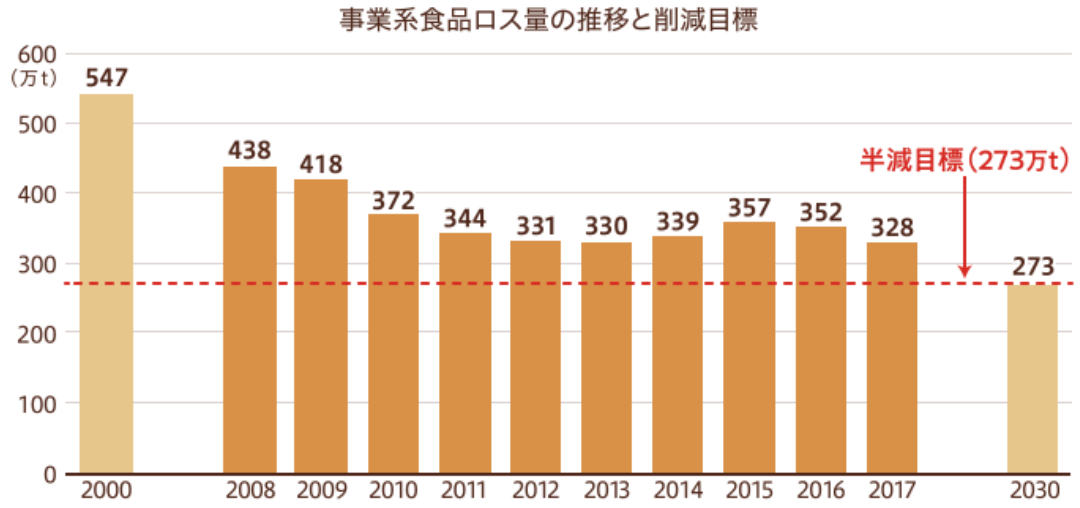


図 1　過去年度の食品ロス量の推移と削減目標値 (出典／食品ロス及びリサイクルをめぐる情勢 (農林水産省))
縦軸:食品ロスの量 (単位:万 t) 横軸:年 赤色の点線:食品ロスの半減目標値 (273 万 t)

農林水産省のアンケート [2] によると、賞味期限内に食べられなかった場合の食品ロスが 70.5 %と 1 位をとっている。また購入後に冷蔵庫や保管場所に入れたまま忘れてしまい破棄せざるおえなくなる場合は 61.1 %と高い数値を得ている。食品ロスにつながる原因のアンケート結果を**図 2** に示す。

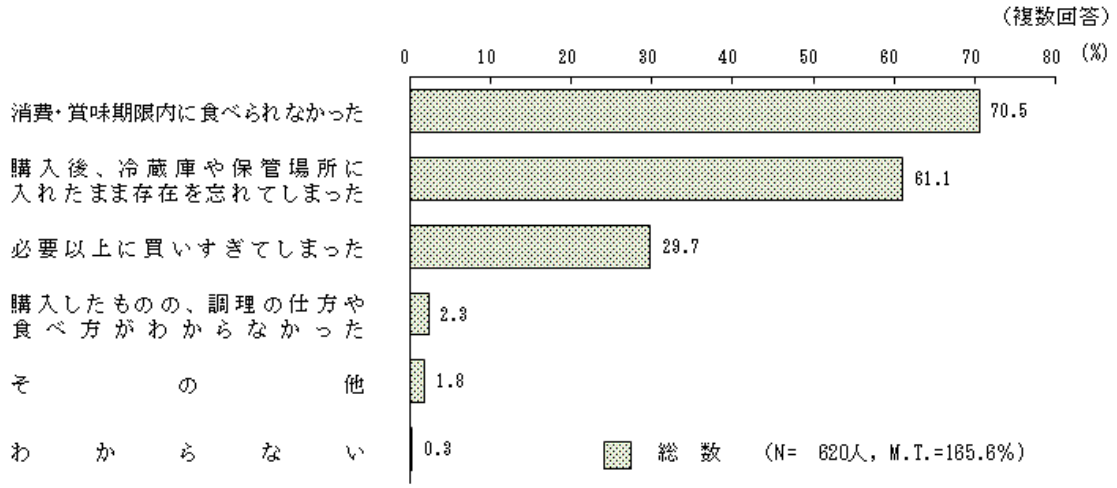


図 2　食品ロスにつながる原因のアンケート結果 (出典／食品ロスの原因 (農林水産省)) 縦:アンケートの質問
横:回答数のパーセンテージ 620 人にアンケートを取った結果。複数回答あり。

3.1 SDGs

SDGs とは Sustainable Development Goals(持続可能な開発目標) [1] の略称である。2015 年 9 月の国連サミットで採択され、国連加盟国 193 カ国が 2016 年から 2030 年の 15 年間で達成するために掲げた目標である。

3.1.1 SDGs の 17 の目標

SDGs は 17 の目標がある。

1. No Poverty
2. Zero Hunger
3. Good Health and Well-Being
4. Quality Education
5. Gender Equality
6. Clean Water and Sanitation
7. Affordable and Clean Energy
8. Decent Work and Economic Growth
9. Industry,Innovation and Infrastructure
10. Reduced Inequality
11. Sustainable Cities and Communities
12. Responsible Consumption and Production
13. Climate Action
14. Life Below Water
15. Life On Land
16. Peace,Justice and Strong Institutions
17. Partnerships for the Goals

3.1.2 日本語訳

1. 貧困をなくそう
2. 飢餓をゼロに
3. すべての人に健康と福祉を
4. 質の高い教育をみんなに
5. ジェンダー平等を表現しよう
6. 安全な水とトイレを世界中に
7. エネルギーをみんなに　そしてクリーンに
8. 働きがいも経済成長も
9. 産業と技術革新の基盤をつくろう
10. 人や国の不平等をなくそう
11. 住み続けられるまちづくりを
12. つくる責任 つかう責任
13. 気候変動に具体的な対策を
14. 海の豊かさを守ろう
15. 陸の豊かさも守ろう

16. 平和と公正をすべての人に
17. パートナーシップで目標を達成しよう

3.2 つくる責任 つかう責任

今回の食品ロスに関係する SDGs は、“つくる責任 つかう責任”である。少ない資源でより多く、より質の高いものを得られるような生産と消費のパターンを作り上げることを目指している。地球上で多くの資源やエネルギーを使用して様々なものを生産し、消費をしている。しかしエネルギーは有限であり大量生産・消費は地球に大きな負担をかけている。必要以上のものを生産し、必要のないものを消費し続けることで地球の有限資源が枯渇する可能性が高まる。これらを防ぐため、生産側は環境や資源を守りながらより少ない資源でより質の高いものを生み出す生産方法の確立と生産工程でのエネルギー消費や廃棄物の発生の抑制を目指し、消費者側は余分な買い物をしない、食材などはできるだけ使い切るなどを意識した生活の実践を目指す。

3.2.1 ターゲット

1. 開発途上国の開発状況や能力を勘案しつつ、持続可能な消費と生産に関する 10 年計画枠組み（10YFP）を実施し、先進国主導の下、全ての国々が対策を講じる。
2. 2030 年までに天然資源の持続可能な管理及び効率的な利用を達成する。
3. 2030 年までに小売・消費レベルにおける世界全体の一人当たりの食料の廃棄を半減させ、 収穫後損失などの生産・サプライチェーンにおける食品ロスを減少させる。
4. 2020 年までに、合意された国際的な枠組みに従い、製品ライフサイクルを通じ、環境上適正な化学物質や全ての廃棄物の管理を実現し、人の健康や環境への悪影響を最小化するため、化学物質や廃棄物の大気、水、土壌への放出を大幅に削減する。
5. 2030 年までに、廃棄物の発生防止、削減、再生利用及び再利用により、廃棄物の発生を大幅に削減する。
6. 特に大企業や多国籍企業などの企業に対し、持続可能な取り組みを導入し、持続可能性に関する情報を定期報告に盛り込むよう奨励する。
7. 国内の政策や優先事項に従って持続可能な公共調達の慣行を促進する。
8. 2030 年までに、人々があらゆる場所において、持続可能な開発及び自然と調和したライフ スタイルに関する情報と意識を持つようにする。
9. 開発途上国に対し、より持続可能な消費・生産形態の促進のための科学的・技術的能力の強化を支援する。
10. 雇用創出、地方の文化振興・産品販促につながる持続可能な観光業に対して持続可能な開発がもたらす影響を測定する手法を開発・導入する。
11. 開発途上国の特別なニーズや状況を十分考慮し、貧困層やコミュニティを保護する形で開発に関する悪影響を最小限に留めつつ、税制改正や、有害な補助金が存在する場合はその環境 への影響を考慮してその段階的廃止などを通じ、各国の状況に応じて、市場のひずみを除去することで、浪費的な消費を奨励する化石燃料に対する非効率な補助金を合理化する。

世界の人口は、2000 年に約 61 億人、2050 年には約 96 億人まで増加し。食料需要は 2000 年に約 45 億トン、2050 年には約 69 億トンまで増加が予想されている。現在世界では 8 億人の人たちが飢餓に苦しんでいる。このまま食料需要量が増加し、また人口も増加すると 2.5 倍の 20 億人が飢餓に苦しむと予想されている。そのためターゲットの 3 つ目が定められている。人口増加を**図 3** に食料需要量を**図 4** に示す。

図 3　世界人口の将来推計の図。2000 年では 61.3 億人だった人口は 2050 年には 95.5 億人にまで増加すると予想されている。縦:人口 (億単位) 横:年数

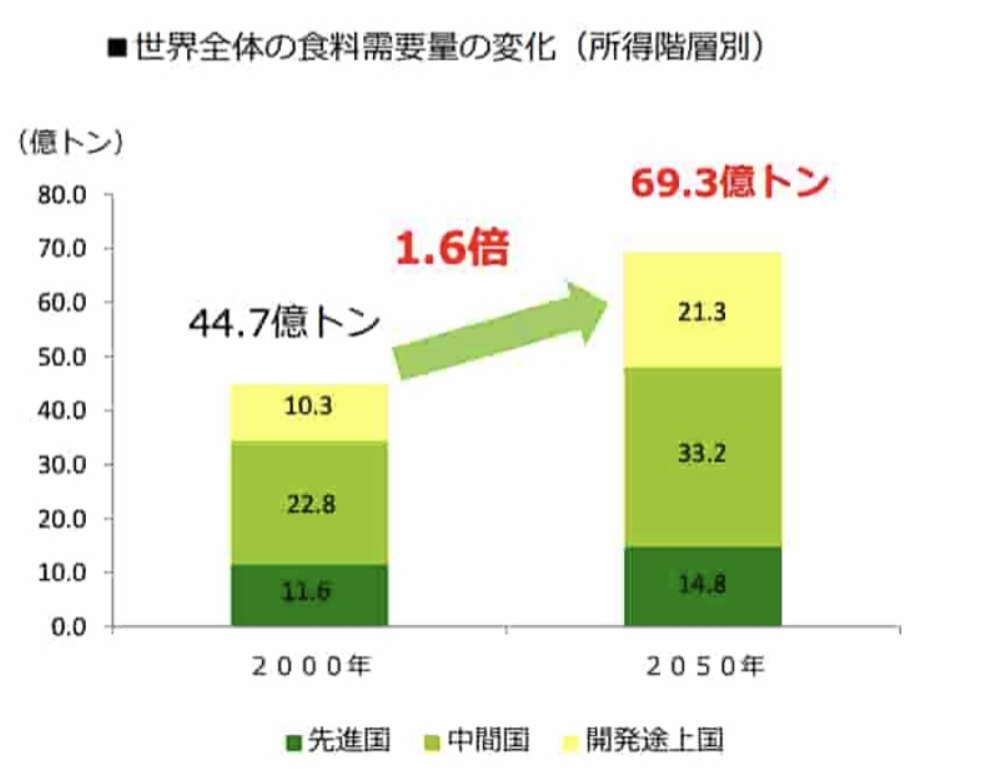


図 4　世界全体の食料需要量の変化。2000 年では人口 61.3 億人に対し 44.7 億トンの食料需要量があった。2050 年には 2000 年の 1.6 倍に人口が増加する予想に伴い、食料需要量も 1.6 倍の 69.3 億トンに増加すると予想されている。縦:食料需要量 (億トン) 横:年数 [3]

4　システムの概要

本研究では、賞味期限を管理し期日に通知をする賞味期限管理システムである。しかし既存のアプリケーションでは写真を撮り自分で賞味期限を入力して管理する作業がある。その点に注目し賞味期限を読み取るだけで自動的に通知が送られてくるシステムを考察した。

本システムでは、スマートフォンで撮影した画像を Raspberry Pi に送り、プログラムを稼働させて賞味期限を読み取ったあと設定日にメールを送信するというシステムである。本システムの流れを**図 5** で示す。

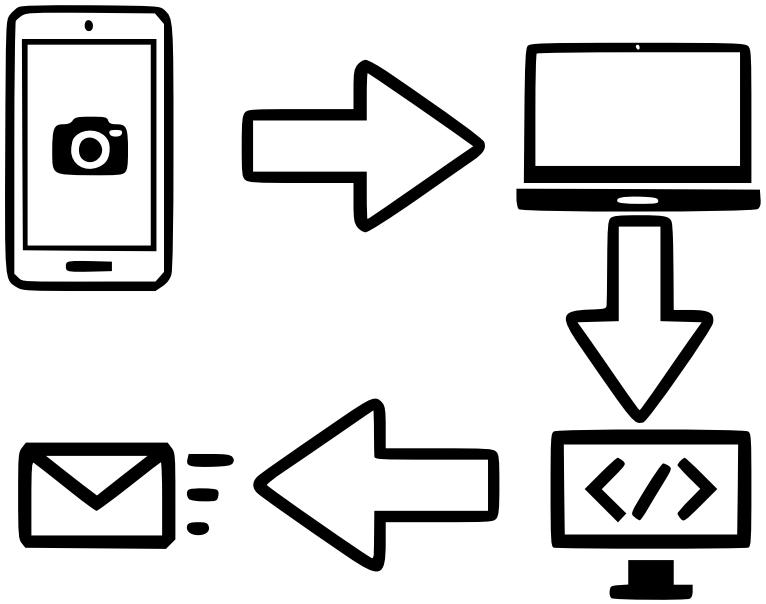


図 5　使用者は、スマートフォンで賞味期限を撮影し Raspberry Pi に送信をする。送信されてきた画像を処理し期日にメールで通知を行う

4.1　Raspberry Pi

本研究で使用する Raspberry Pi とは、ARM プロセッサを搭載したシングルボードコンピュータである。シングルボードコンピュータとはパソコンとして最低限の機能が備わった入出力装置である。HDMI や USB、無線 LAN なども搭載されており拡張性に優れている。当初の目的としてプログラミング教育を行うため、安価で提供できるように開発された。またプログラミングを行うにあたってデフォルトで総合開発環境が備わっているため、Windows や Mac での手間のかかるプログラミング環境を作成する必要がなく誰でも簡単にプログラミングを行うことができる。また、Raspberry Pi の登場により IoT 開発が身近になった。IoT(Internet of Things) とは「モノのインターネット」という意味合いで「身の回りのあらゆるモノがインターネットにつながる」ということを指す。プログラムを常に動かすためには、少ない電力量でプログラムを動かし続けることができる Raspberry Pi が適していると考えた。

また Raspberry Pi の人気は衰えることなく、2021 年の時点で 4000 万台以上が出荷されている。Raspberry Pi の売上グラフを**図 6** に示す。

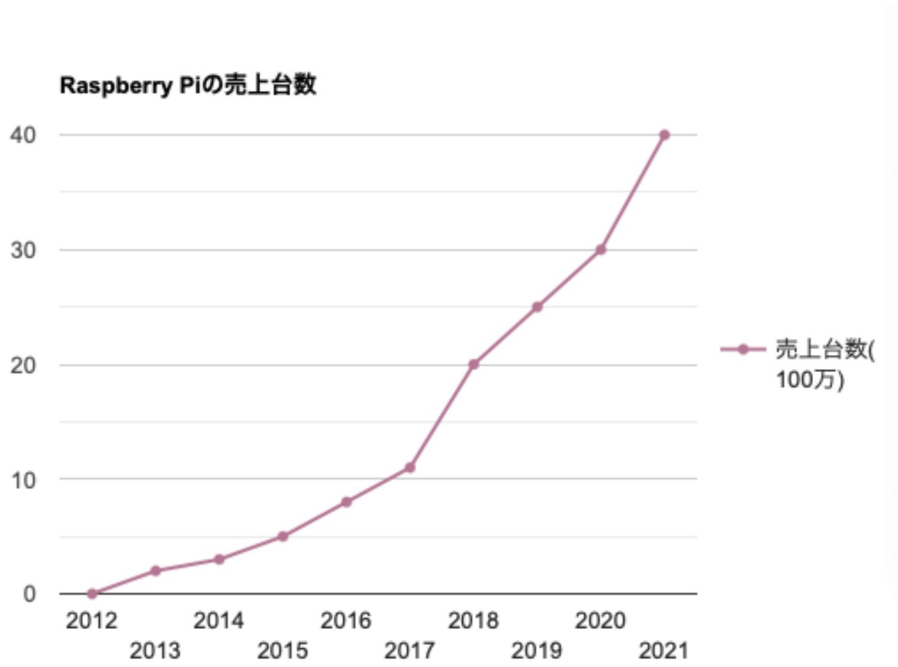


図 6　Raspberry Pi　 2021 年までの売上台数 [4]　縦:売上台数 (100 万単位) 横:年

4.2　OCR(Optical Character Reader)

本研究で使用する OCR とは、光学文字認識と呼ばれ紙媒体や画像などに書かれている文字を認識してデジタル化する業務効率ツールとして注目されている。導入例としては、会社の経理や会計業務において請求書や伝票などの入力作業を OCR を用いて効率化する。手書き資料を読み取り文書ファイルとしてデジタル化し保存をすることで物理的な容量を削減し、更にデータ内でワード検索をすることが可能になり必要な資料などを素早く見つけ出すことが可能になるなど業務の効率化として導入されている。その他にも自動車ナンバー自動読み取り、交通標識認識システムなどにも使用されている。OCR の原点は 1914 年まで遡る。電子技術の拡張、視覚障害者が文字を読むための機械の開発。この二つの問題にまつわる活動から誕生した。

OCR は誰でも簡単の使うことができる。例えば Google Drive に OCR が導入されている。画像や書籍などを Google Drive にアップロードし、Google ドキュメントで開くことで自動的に OCR でテキスト化される。実際に Google Drive の OCR を使った時の使用画像を**図 7** に結果を**図 8** にそれぞれ示す。

あいうえお
かきくけこ
さしすせそ
たちつてと
なにぬねの
はひふへほ
まみむめも
やゆよ
らりるれろ
わをん

図 7　実際に使用したテキストが記載されている画像

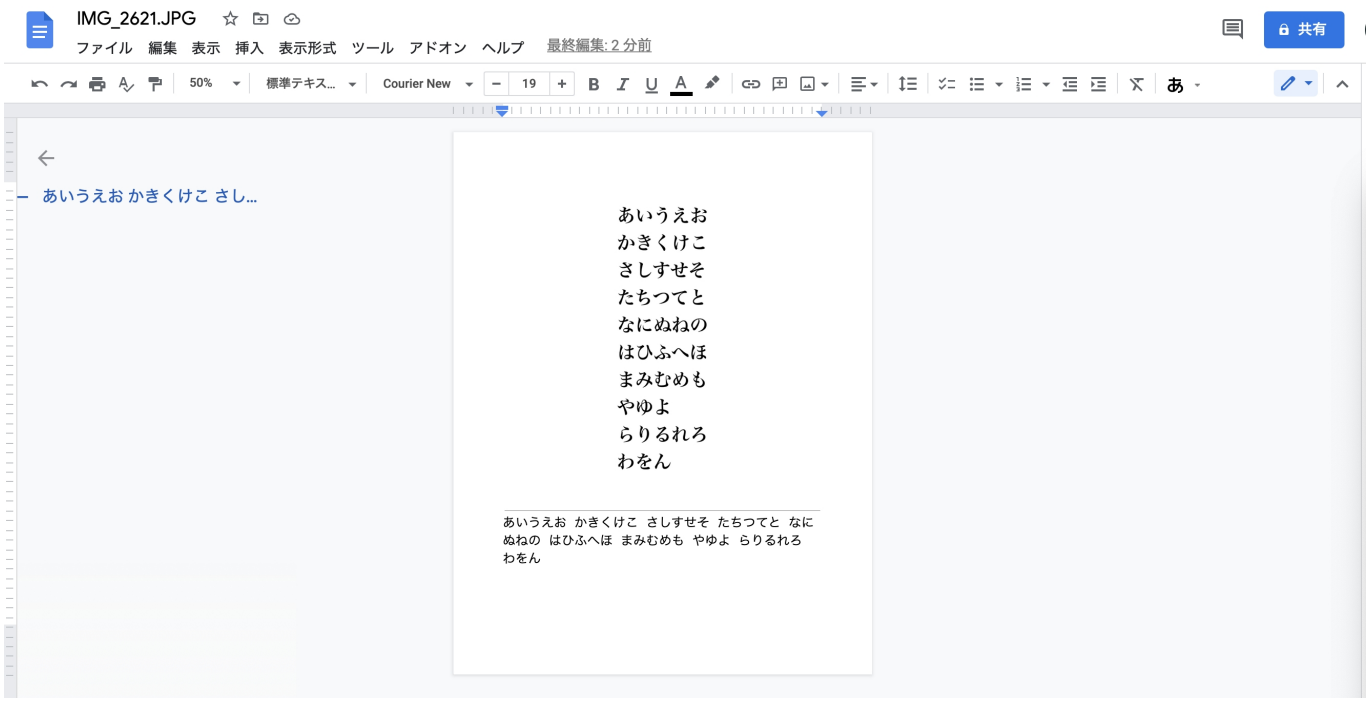


図 8　Google Drive で OCR を使用した例。読み取ることで実際にテキストとして扱える。ここからコピーして文章としても扱えるようになる。

4.3 システムの詳細

本研究の開発システムは、画像から賞味期限を取得し賞味期限日がくると Gmail からメール通知が届くシステムである。プログラムの言語は Python を用いた。

プログラムは画像を読み取り JSON ファイルを作成するプログラム、JSON ファイルに記載されている期日と現在の日付が同じの場合メールを送信するプログラムの二つとこの両方を定期的に稼働させるプログラムの 3 つで構成されている。

4.4 賞味期限の取得

食品の画像から賞味期限を読み取る際はスマートフォンのカメラを使って撮影した。スマートフォンで撮影された画像は一度パソコンに送信し、パソコンから Raspberry Pi へ送信する。また賞味期限を取得するという点から食品の全体ではなく賞味期限のみを撮影した画像を使用する。

4.4.1 取得方法

画像から賞味期限を取得するには OCR(Optical Character Reader) を用いて賞味期限を取得するため、pyocr を用いた。OCR は文字や数字、記号を読み取ることができるが本システムに必要なのは賞味期限のみなため、数字だけを読み取るように設定した。OCR の読み取り設定について**表 1** に示す。

表 1　OCR 読み取り設定。OCR を使う際にそれぞれ設定をすることで様々な用途に用いることができる。

TextBuilder	文字列を認識
WordBoxBuilder	単語単位で文字認識 BoundingBox
LineBoxBuilder	行単位で文字認識 BoundingBox
DigitBuilder	数字/記号を認識
DigitlineBoxBuilder	数字/記号を認識 BoundingBox

今回は数値のみを読み取りたいため“DigitBuilder”に設定した。また食品などに記載されている賞味期限の記載方法が“/”や“-”などの記号で区切られている場合があるので読み取った数値はそれぞれの桁の数値に変換する。現在の日付は datetime モジュールで取得するが取得した数値は文字列であるため、読み取った数値も文字列に変換する。

賞味期限の記載には種類があるため全ての記載に対応できるようにした。例えば‘2021,11,11’や‘2021,01,01’などの 8 桁記載。‘21,1,1’などの西暦が 2 桁、日付が一桁の場合や 0 がつかない場合の 4 桁記載などである。また読み取った賞味期限は、通知する 3 日前に設定するため 3 日分減算をする。3 日前に定めた理由としては、人はモノを記憶してから 1 週間で 77 %を忘れてしまうという [5] から 3 日と定めた。また期日当日だと余裕がなくてをつけられないと考えた。

4.4.2 JSON ファイルの書き出し

Python から Gmail を送る際にリマインダー機能として JSON ファイルに賞味期限を保存して行なった。JSON ファイルとして保存しておくことでリマインダー機能となり、後にメール作成プログラムによって保存されている日付に読み込むことができる。作成された JSON ファイルを**図 9** にフローチャートを**図 10** にそれぞれ示す。

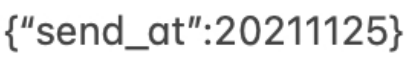


図 9　実際に作成された JSON ファイル。日付を JSON ファイルとして保存しておくことで書き込まれた日付にメールを送信するプログラムを作動させることができる。

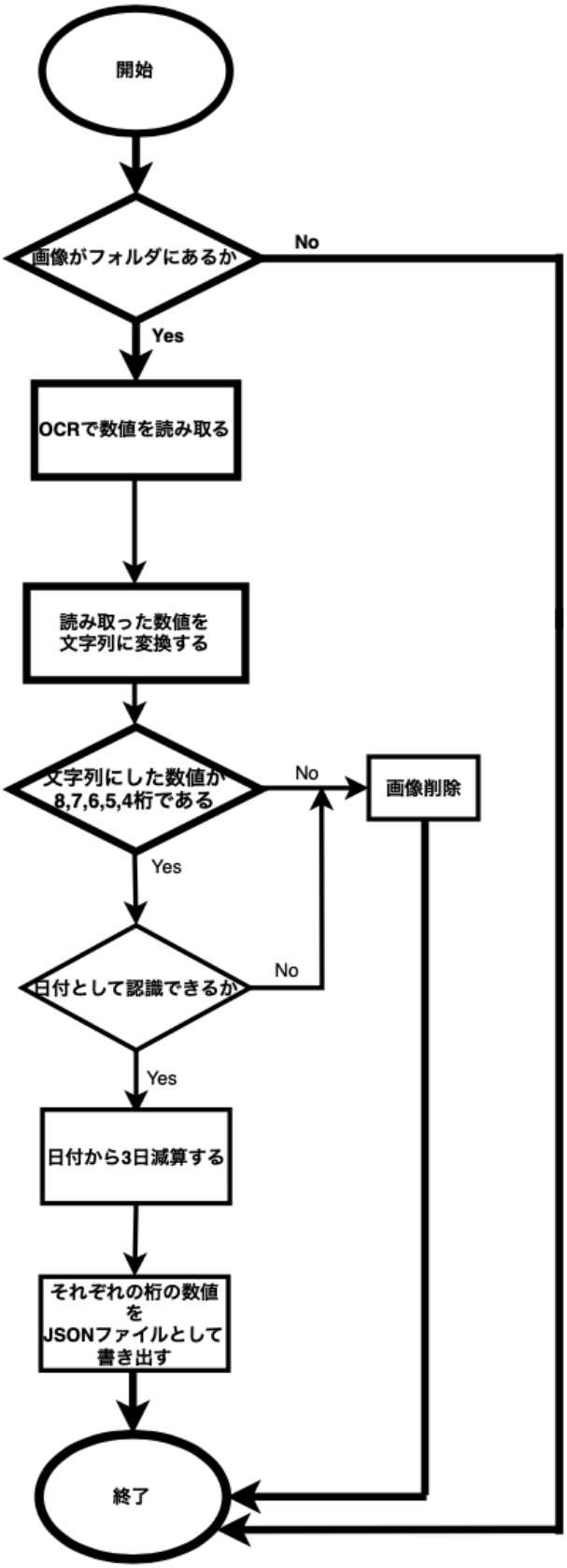


図 10　OCR で画像を読み取り JSON ファイルを書き出すプログラムのフローチャート 数値を読み取った時に賞味期限は最大 8 桁から最小 4 桁のため、桁数で判別する。次に読み取った数値を日付として読み取れるかを判別し、読み取れた場合日付から 3 日減算する。最後にリマインダーの役割の JSON ファイルを書き出す。

4.5　メールの作成

本研究のシステムの通知は Gmail を用いて通知した。メールアドレスは大学用の ge アドレスを使用した。賞味期限の文字列が保存された JSON ファイルを読み取り、現在の日付と賞味期限日から 3 日減算した日付が同日ならメールを送信する。またストレージの圧迫を防ぐためメールを送信すると JSON ファイルは削除されるようになっている。メール文は、賞味期限のものがあることを知らせ、読み取る際に撮影した画像を添付したメール文になっている。フローチャートを**図 11** に、実際に届くメールを**図 12** にそれぞれ示す。

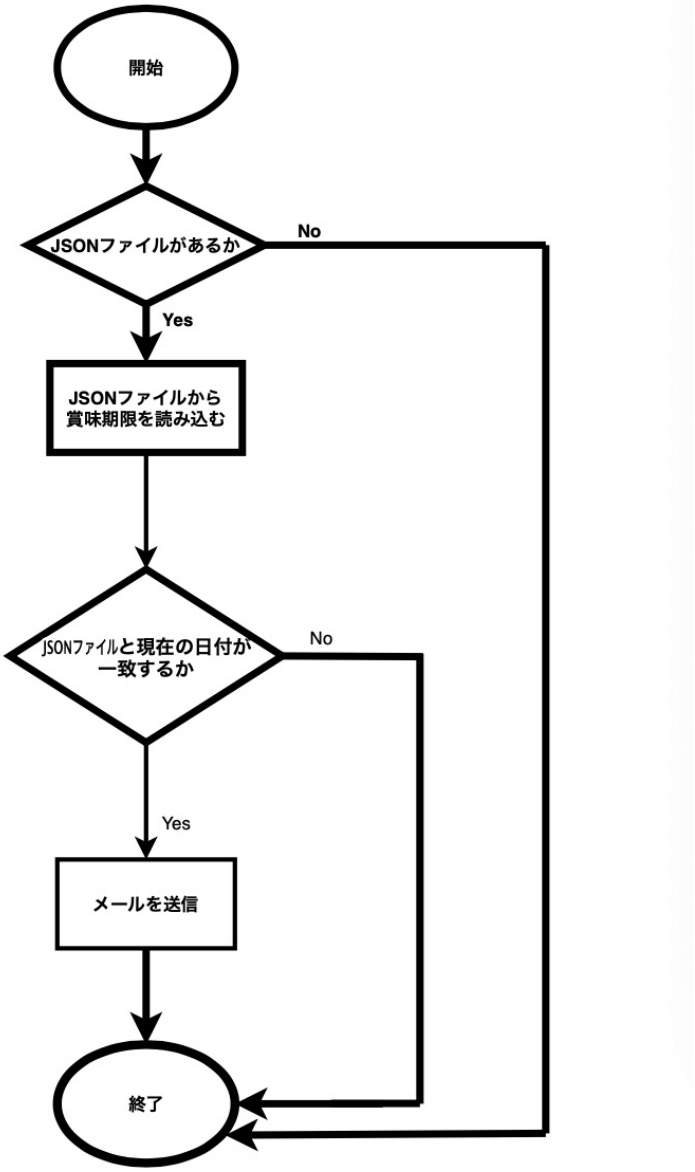


図 11　メールを作成するプログラムのフローチャート 前記のプログラムで書き出された JSON ファイルにある日付と現在の日付を比較し、一致した場合読み込んだ際に使用した画像を添付しメールを送信する。メール送信後、画像と JSON ファイルを削除する。

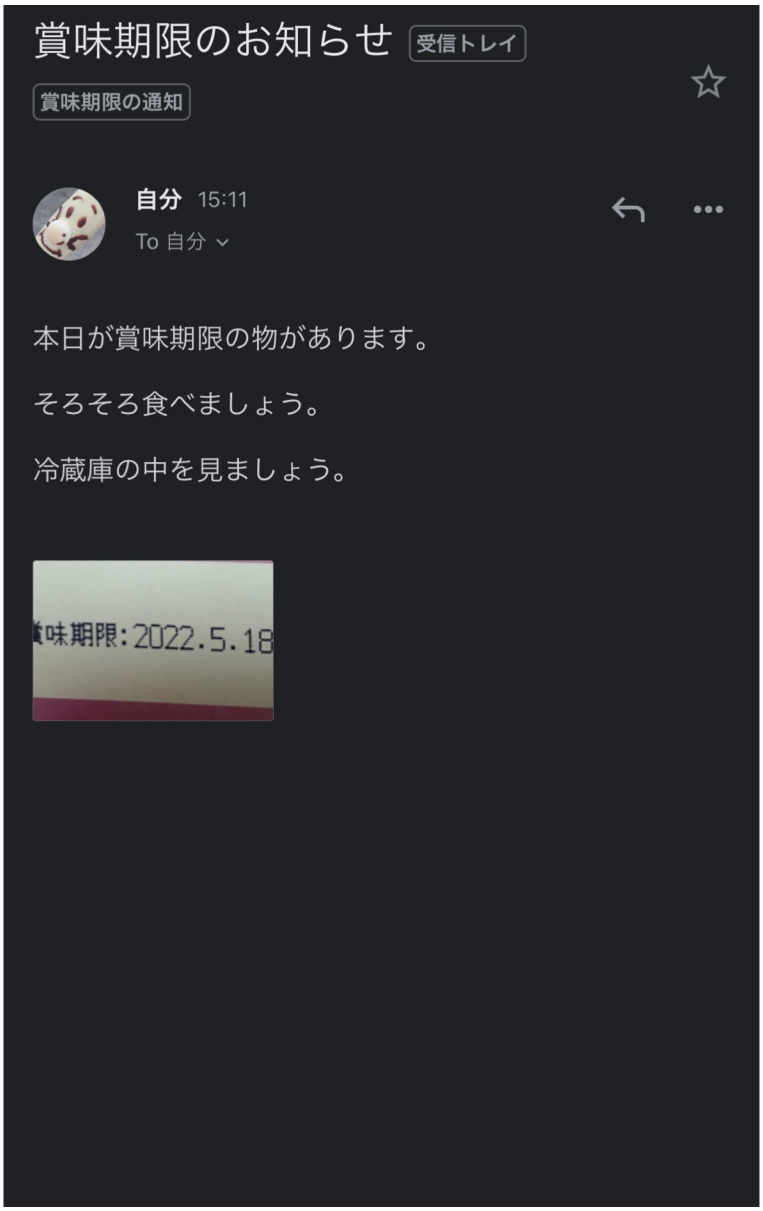


図 12　Raspberry Pi から届いた Gmail。メールの件名に“賞味期限のお知らせ”とし、実際に読み取る際に使用した画像が添付されて送られてくる。

4.6　プログラムを定期的に動かす

画像を送信すると自動的に賞味期限を読み取りメールを送信されるようにする。schedule モジュールをインストールした。賞味期限を読み取るプログラムは、複数の画像を短時間で送信することを考慮し 10 秒ごとに実行されるようにした。また、Gmail を送信するプログラムは、賞味期限から 3 日前の毎朝午前 9 時に実行されるようにした。

5　検証

本システムを導入した際に賞味期限をどれだけ正確に読み取ることができるのか検証を行なった。食品には様々な記載方法がある。また、写真の撮影方法にも変化があると考えた。

5.1　検証方法

賞味期限が記載の写真を 5 枚用意した。商品の全体画像ではなく、賞味期限のみが写っている写真を使用する。また実線で記載されているもの、ドットで記載されているもの、他の文字が記載されているもの、逆さまで撮影されたものでも正確に読み取れているかどうかを検証する。

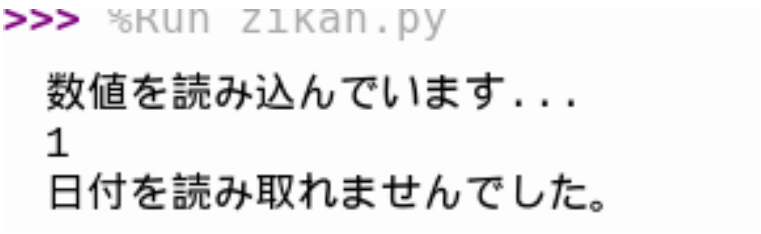
5.1.1　検証するもの

1. ペットボトルに記載された細く書かれた賞味期限
2. 実線の賞味期限の近くに漢字が記載されているもの
3. 逆さまに書かれた賞味期限
4. ドットで書かれた賞味期限
5. 賞味期限以外に英語が書かれているもの
6. 商品名と賞味期限を写したもの

5.2　検証結果

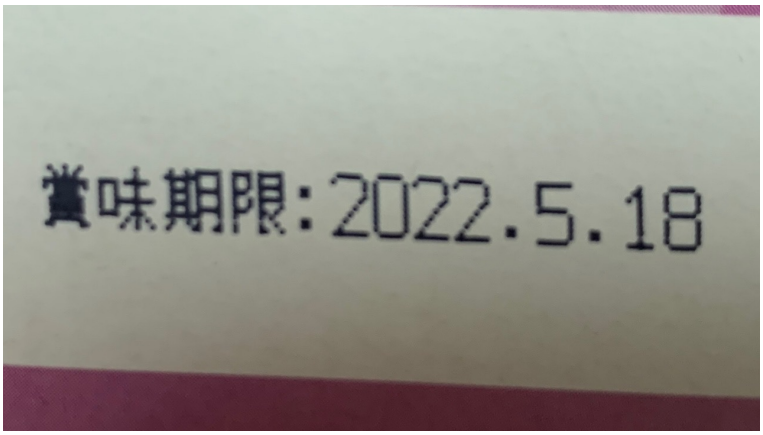
5.2.1　ペットボトルに記載された細く書かれた賞味期限

↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓結果↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓



うまく読み取ることができなかった。

5.2.2　実線の賞味期限の近くに漢字が記載されているもの

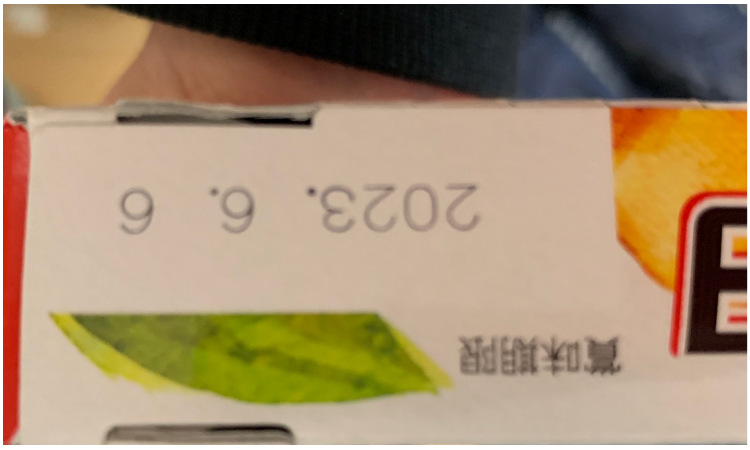


↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓結果↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓

```
>>> %Run zikan.py
数値を読み込んでいます...
2022518
```

うまく読み取ることができた。賞味期限という文字は読み込まず、数字の賞味期限だけを読み取ることがきた。

5.2.3　逆さまに書かれた賞味期限



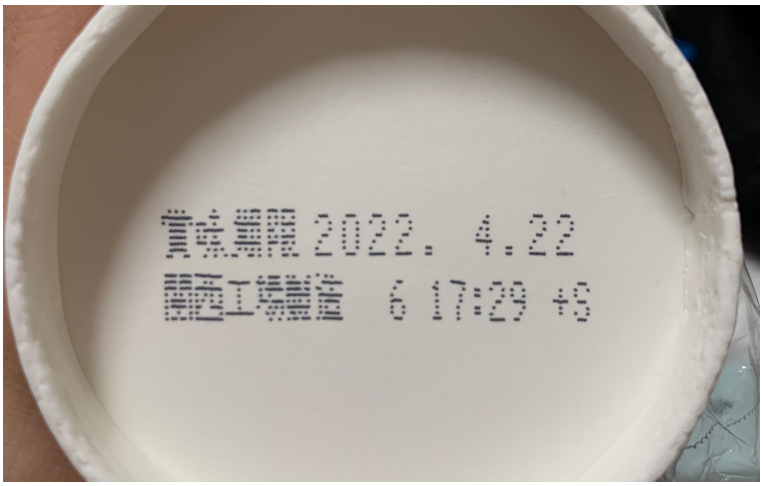
↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓結果↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓

```
>>> %Run zikan.py
  数値を読み込んでいます...

  日付を読み取れませんでした。
```

逆さまのものは数字として読み取ることができなかった。6 を反対にして 9 と視覚的には見えるが 9 としても読み取ることはできなかった。

5.2.4 ドットで書かれた賞味期限

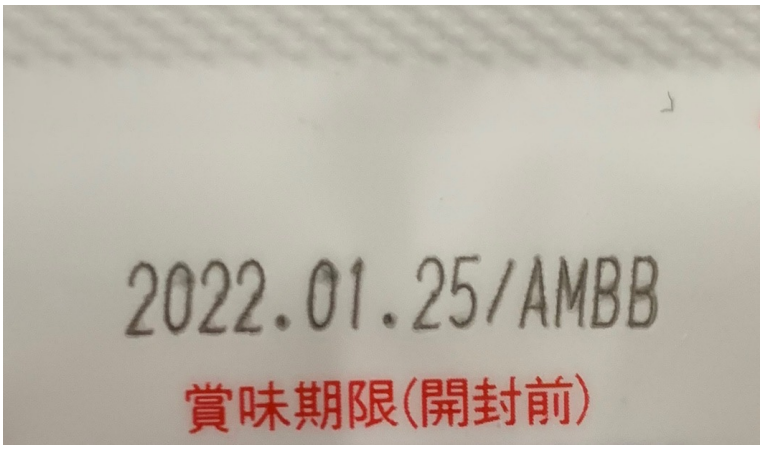


↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓結果↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓

```
>>> %Run zikan.py
  数値を読み込んでいます...
  7433242373982027 3
  日付を読み取れませんでした。
```

ドットで書かれたものは、書いているものとは違う数字として読み取られてしまい、他の文字も数字として読み込まれてしまう。

5.2.5 アルファベットなどが後ろに続いているもの



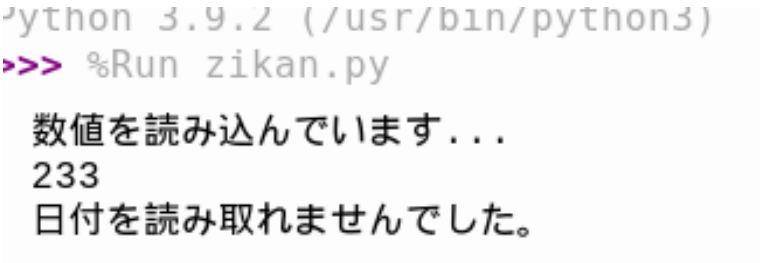
↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓結果↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓

```
>>> %Run zikan.py
数値を読み込んでいます...
202201257
日付を読み取れませんでした。
```

賞味期限以外に記載されている賞味期限は、数字はうまく読み取ることができ“賞味期限 (開封前)”という部分を除くことができたが、アルファベットを数字として読み込んでしまう。

5.2.6 商品名と賞味期限を写したもの

↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓結果↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓



全体像を写すと目視では賞味期限は確認できるものも読み取ることができなかった。

6　結果のまとめ

検証の結果、賞味期限がはっきり記載がされていないと読み取ることができないとわかった。特にドットで記載されているものは、認識することができないこともある。漢字は文字として正確に除くことができたが、アルファベットは文字として除くことができず数字として読み込まれてしまうことがわかった。

OCR の精度が高くはないのでどんなものでも読み取ることができるようにするのが改善点である。またアプリケーションとしてはまだ不便であることが多いため様々な下記の改善が必要である。

6.1　問題点

6.1.1　検証結果の問題点

1. 実線で記載されているものしか読み取ることができない
   様々な食品がある中で全てが同じように賞味期限が記載されているわけではない。そのため実線以外で記載されているものがある場合読み取ることができない。
2. アルファベットを数字として読み取ってしまう
   OCR の設定では、数字のみを読み取る設定にしているが数字を読み取る精度を上げるために英語設定にしている。そのためアルファベットも数字として間違って読み取ってしまう。
3. 賞味期限のみを中心に撮影をしないとうまく読み取れない
   設定上、数字を読み取るようにはなっているが食品全体を撮影すると他の数字も読み取ってしまい賞味期限を取得することができない。また字が小さいと数字としても読み取ることができなかった。

6.1.2　その他の問題点

1. 賞味期限日が来る前に食品を食べ終わっても通知が来る
   賞味期限を通知する前に食品を食べた場合でもファイルに JSON ファイルが残ったままになっているため期限日が来た時には通知されてしまう。
2. スマートフォンで写真を撮り Raspberry Pi に送るという手間がある
   撮影した画像は Raspberry Pi に送信しないと何も起動しないため手間がかかってしまう。
3. 日付が重複した場合、上書きされてしまう
   送信された画像や作成された JSON ファイルは日付で管理されているため、同日の食品がある場合上書きされてしまう。
4. 野菜や魚など賞味期限の記載がないものは管理できない
   システム上、賞味期限を読み取ることで通知するシステムのため賞味期限の記載がない食品は使用することができない。
5. メールを読むだけでどの食品なのか一目でわからない。
   本研究では、OCR で賞味期限を読み取るという点を重視しているため添付されて送られてくるメール文には賞味期限のみの画像が送られてくる。それを一目でどの食品かを確認することが難しい。

6.2　現在考えられる改善策

- 賞味期限日が来る前に食品を食べ終わった際にも通知が来てしまう問題点として、WEB ページを使用することで改善されると考える。活用法としては、画像が送られてきた食品の賞味期限を読み取り、JSON ファイルとして書き出した際に WEB ページに送信されてきた画像と日付をアップロードし、消費した際にい

つでも削除ができると同時に JSON ファイルも削除できるようにすれば改善されると考える。

- スマートフォンで写真を撮り Raspberry Pi に送るという手間がある問題点としては、まずアプリケーションの動きは定まっているため、スマートフォン一つで完結できるようにしなくてはならない。そこで次はアプリケーションの開発を進めなければいけない。
- メールを読むだけでどの食品なのか一目でわからないという問題点は、食品の全体画像から文字や数字を全て読み取り、その中から賞味期限だけを抽出するようにできれば現在のシステムより利便性が向上すると考える。

7　結論

本研究では、OCR を用いた賞味期限管理システムの開発を行なった。その結果、OCR 用いた賞味期限管理システムを作成することができたが、まだ便利なものとはいえない。しかし、OCR を用いることで日付を自動で読み取りリマインドすることが可能であるとわかった。

謝辞

本研究の研究作業を進めていく上で大垣斉准教授からご協力及びご指導頂き、深く感謝致します。また、情報教育システム研究室のメンバー及び卒業生の方々に深く感謝の意を表します。

参考文献

[1] Sdgs とは？英語と日本語で 17 項目のゴールを学ぼう！ — 語学をもっと身近に「ecc フォリラン！」公式サイト. https://foreignlang.ecc.co.jp/know/k00118d/.

[2] 図 7 - 6 食品ロスの原因：農林水産省. https://www.maff.go.jp/j/syokuiku/ishiki/h29/zuhyou/z7-6.html.

[3] 世界の将来人口及び食料需要. https://www.maff.go.jp/tokai/kikaku/jikyu/lecture/attach/pdf/20180613-7.pdf. (Accessed on 02/12/2022).

[4] Raspberry pi - wikipedia. https://ja.wikipedia.org/wiki/Raspberry_Pi.

[5] 人は 1 日のうちに半数以上の記憶がなくなる!?「忘却曲線」を理解して営業に活かそう | ferret. https://ferret-plus.com/4951.

付録 A　ソースコード

A.1　OCR を使って賞味期限を読み取るプログラム

Listing 1　foo

```
import pyocr as ocr
import pyocr.builders
from PIL import Image
import os
import json
import time
import re
import datetime
import os.path



def yomitori():


    if os.path.exists('/home/pi画像//ocr_test.jpeg') == True :
        print("数値を読み込んでいます...")
        img_path = "画像..//ocr_test.jpeg"

        tools = pyocr.get_available_tools()
        if len(tools) == 0:
            sys.exit(1)



        tool = tools[0]
        res = tool.image_to_string(Image.open(img_path), lang = "eng", builder = pyocr.bui

        result = re.sub(r"\D", "", res)

        if len(result) == 8 :
            result1 = datetime.datetime.strptime(result,'%Y%m%d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        elif len(result) == 7:
            result1 = datetime.datetime.strptime(result,'%y%-m%-d')
```

```
        da1 = result1 + datetime.timedelta(days=-3)
        da2 = da1.strftime('%Y%m%d')
    elif len(result) == 6:
        result1 = datetime.datetime.strptime(result,'%Y%-m%-d')
        da1 = result1 + datetime.timedelta(days=-3)
        da2 = da1.strftime('%Y%m%d')
    elif len(result) == 5 or len(result) == 4:
        result1 = datetime.datetime.strptime(result,'%y%-m%-d')
        da1 = result1 + datetime.timedelta(days=-3)
        da2 = da1.strftime('%Y%m%d')
    else:
        return
    resultf = re.sub(r"\D", "", da2)
    print(resultf)

    i = resultf

    path1 = '画像..//ocr_test.jpeg'
    path2 = '画像..//{}.jpeg'.format(i)
#ファイル名を読み込んだ数値に変更する
    os.rename(path1,path2)

#数値を読み込んだらファイルを出力するが、それ以外は削除する。JSON
    if len(result) == 8 or len(result) == 7 or len(result) == 6 or len(result) == 5 or




        str = {

                'send_at':resultf
            }
        with open('/home/pi/json_folder/{}.json'.format(i),'w') as f:
         json.dump(str,f)

         os.remove('/home/pi画像//{}.jpeg'.format(i))
```

```
    else:
        os.remove('/home/pi画像//{}.jpeg'.format(i))
        print("日付を読み取れませんでした。")

#----------------------------------------JPG----------------------------------------

elif os.path.exists('/home/pi画像//ocr_test.JPG') == True :
        print("数値を読み込んでいます...")
        img_path = "画像..//ocr_test.JPG"

        tools = pyocr.get_available_tools()
        if len(tools) == 0:
            sys.exit(1)


        tool = tools[0]
        res = tool.image_to_string(Image.open(img_path), lang = "eng", builder = pyocr.

        result = re.sub(r"\D", "", res)
        print(result)
        if len(result) == 8 :
            result1 = datetime.datetime.strptime(result,'%Y%m%d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        elif len(result) == 7:
            result1 = datetime.datetime.strptime(result,'%y%-m%-d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        elif len(result) == 6:
            result1 = datetime.datetime.strptime(result,'%Y%-m%-d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        elif len(result) == 5 or len(result) == 4:
            result1 = datetime.datetime.strptime(result,'%y%-m%-d')
```

```
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        else:
            return
        resultf = re.sub(r"\D", "", da2)
        print(resultf)

        i = resultf

        path1 = '画像..//ocr_test.JPG'
        path2 = '画像..//{}.JPG'.format(i)
#ファイル名を読み込んだ数値に変更する
        os.rename(path1,path2)

#数値を読み込んだらファイルを出力するが、それ以外は削除する。JSON
        if len(result) == 8 or len(result) == 7 or len(result) == 6 or len(result) == 5




            str = {

                    'send_at':resultf
                }
            with open('/home/pi/json_folder/{}.json'.format(i),'w') as f:
             json.dump(str,f)
      #       os.remove('/home/pi画像//{}.JPG'.format(i))




        else:
            os.remove('/home/pi画像//{}.JPG'.format(i))
            print("日付を読み取れませんでした。")
```

```
#----------------------------------------------------------jpg----------------------------------------------------------

    elif os.path.exists('/home/pi画像//ocr_test.jpg') == True :
        print("数値を読み込んでいます...")
        img_path = "画像..//ocr_test.jpg"

        tools = pyocr.get_available_tools()
        if len(tools) == 0:
            sys.exit(1)

        tool = tools[0]
        res = tool.image_to_string(Image.open(img_path), lang = "eng", builder = pyocr.

        result = re.sub(r"\D", "", res)

        if len(result) == 8 :
            result1 = datetime.datetime.strptime(result,'%Y%m%d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        elif len(result) == 7:
            result1 = datetime.datetime.strptime(result,'%y%-m%-d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        elif len(result) == 6:
            result1 = datetime.datetime.strptime(result,'%Y%-m%-d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        elif len(result) == 5 or len(result) == 4:
            result1 = datetime.datetime.strptime(result,'%y%-m%-d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        else:
           return
        resultf = re.sub(r"\D", "", da2)
        print(resultf)

        i = resultf

        path1 = '画像..//ocr_test.jpg'
```

```
            path2 = '画像..//{}.jpg'.format(i)
#ファイル名を読み込んだ数値に変更する
            os.rename(path1,path2)

#数値を読み込んだらファイルを出力するが、それ以外は削除する。JSON
            if len(result) == 8 or len(result) == 7 or len(result) == 6 or len(result) == 5




                str = {

                        'send_at':resultf
                    }
                with open('/home/pi/json_folder/{}.json'.format(i),'w') as f:
                 json.dump(str,f)
                 os.remove('/home/pi画像//{}.jpg'.format(i))




            else:
                os.remove('/home/pi画像//{}.jpg'.format(i))
                print("日付を読み取れませんでした。")




#----------------------------------------------------------png----------------------------------------------------------


        elif os.path.exists('/home/pi画像//ocr_test.png') == True :
            print("数値を読み込んでいます...")
            img_path = "画像..//ocr_test.png"

            tools = pyocr.get_available_tools()
            if len(tools) == 0:
                sys.exit(1)
```

```
        tool = tools[0]
        res = tool.image_to_string(Image.open(img_path), lang = "eng", builder = pyocr.

        result = re.sub(r"\D", "", res)

        if len(result) == 8 :
            result1 = datetime.datetime.strptime(result,'%Y%m%d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        elif len(result) == 7:
            result1 = datetime.datetime.strptime(result,'%y%-m%-d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        elif len(result) == 6:
            result1 = datetime.datetime.strptime(result,'%Y%-m%-d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        elif len(result) == 5 or len(result) == 4:
            result1 = datetime.datetime.strptime(result,'%y%-m%-d')
            da1 = result1 + datetime.timedelta(days=-3)
            da2 = da1.strftime('%Y%m%d')
        else:
           return
        resultf = re.sub(r"\D", "", da2)
        print(resultf)

        i = resultf

        path1 = '画像..//ocr_test.png'
        path2 = '画像..//{}.png'.format(i)
#ファイル名を読み込んだ数値に変更する
        os.rename(path1,path2)

#数値を読み込んだらファイルを出力するが、それ以外は削除する。JSON
        if len(result) == 8 or len(result) == 7 or len(result) == 6 or len(result) == 5



            str = {
```

```
            'send_at':resultf
        }
    with open('/home/pi/json_folder/{}.json'.format(i),'w') as f:
     json.dump(str,f)
    os.remove('/home/pi画像//{}.png'.format(i))
```

```
else:
    os.remove('/home/pi画像//{}.png'.format(i))
    print("日付を読み取れませんでした。")
```

A.2 Gmail を送信するプログラム

Listing 2 hoge

```
import smtplib, ssl
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email.mime.application import MIMEApplication
from email import encoders
import json
import datetime
import time
import os
import os.path
import glob



def mail():現在の日にち

    #
    dt_now = datetime.datetime.now()年月日のみを表示する
    #
    date_now = 'd_today.year,.,d_today.moth,.,d_today.day'メールを送る宛先&

    #にログインgmail
```

```
    gmail_account = "s18h047@ge.osaka-sandai.ac.jp"
    gmail_password = "kenta691713"
    mail_to = "s18h047@ge.osaka-sandai.ac.jp"年月日を文字列に変換

    #
    td = datetime.date.today()
    td2 = td.strftime('%Y%-m%-d')
    td3 = td.strftime('%y%-m%-d')
    td4 = td.strftime('%y%m%d')
    td5 = td.strftime('%Y%m%d')
    #ファイルのを読み込むJSONsend_at
    if os.path.exists('/home/pi/json_folder/{}.json'.format(td2)):

        json_open = open('/home/pi/json_folder/{}.json'.format(td2),'r')
        json_load = json.load(json_open)

    elif os.path.exists('/home/pi/json_folder/{}.json'.format(td3)):

        json_open = open('/home/pi/json_folder/{}.json'.format(td3),'r')
        json_load = json.load(json_open)

    elif os.path.exists('/home/pi/json_folder/{}.json'.format(td4)):

        json_open = open('/home/pi/json_folder/{}.json'.format(td4),'r')
        json_load = json.load(json_open)

    elif os.path.exists('/home/pi/json_folder/{}.json'.format(td5)):

        json_open = open('/home/pi/json_folder/{}.json'.format(td5),'r')
        json_load = json.load(json_open)

    else:

        print("本日の賞味期限のものはありません")
        return

#件名と本文
    subject = "賞味期限のお知らせ"
    body = """
    <html>
```

```
␣␣␣␣␣␣␣␣␣<body>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<p賞味期限がきれるまで残り３日のものがあります。></p>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<pそろそろ食べましょう。></p>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<p冷蔵庫の中を見ましょう。></p>

␣␣␣␣␣␣␣␣␣</body>
␣␣␣</html>"""

    filepath = '/home/pi画像//{}.JPG'.format(td2)
    filename = os.path.basename(filepath)



    msg = MIMEMultipart()
    msg["Subject"] = subject
    msg["To"] = mail_to
    msg["From"] = gmail_account
    msg.attach(MIMEText(body,"html"))







    if json_load['send_at'] == td2:

            filepath = '/home/pi画像//{}.JPG'.format(td2)
            filename = os.path.basename(filepath)

            with open(filepath,"rb") as f:
                mb = MIMEApplication(f.read())
                mb.add_header('Content-Disposition','attachment',filename=filename)
                msg.attach(mb)

            server = smtplib.SMTP_SSL("smtp.gmail.com",465,context = ssl.create_default_con
            server.login(gmail_account,gmail_password)
            server.send_message(msg)
            os.remove('/home/pi/json_folder/{}.json'.format(td2))
            print("メールの送信が完了しました")
            json_load=[]
            os.remove('/home/pi画像//{}.JPG'.format(td2))
```

```
elif json_load['send_at'] == td3:

        filepath = '/home/pi画像//{}.JPG'.format(td3)
        filename = os.path.basename(filepath)

        with open(filepath,"rb") as f:
            mb = MIMEApplication(f.read())
            mb.add_header('Content-Disposition','attachment',filename=filename)
            msg.attach(mb)

        server = smtplib.SMTP_SSL("smtp.gmail.com",465,context = ssl.create_default_con
        server.login(gmail_account,gmail_password)
        server.send_message(msg)
        os.remove('/home/pi/json_folder/{}.json'.format(td3))
        print("メールの送信が完了しました")
        json_load=[]
        os.remove('/home/pi画像//{}.JPG'.format(td3))

elif json_load['send_at'] == td4:


        filepath = '/home/pi画像//{}.JPG'.format(td4)
        filename = os.path.basename(filepath)

        with open(filepath,"rb") as f:
            mb = MIMEApplication(f.read())
            mb.add_header('Content-Disposition','attachment',filename=filename)
            msg.attach(mb)

        server = smtplib.SMTP_SSL("smtp.gmail.com",465,context = ssl.create_default_con
        server.login(gmail_account,gmail_password)
        server.send_message(msg)
        os.remove('/home/pi/json_folder/{}.json'.format(td4))
        print("メールの送信が完了しました")
        json_load=[]
        os.remove('/home/pi画像//{}.JPG'.format(td4))

elif json_load['send_at'] == td5:
```

```
        filepath = '/home/pi画像//{}.JPG'.format(td5)
        filename = os.path.basename(filepath)

        with open(filepath,"rb") as f:
            mb = MIMEApplication(f.read())
            mb.add_header('Content-Disposition','attachment',filename=filename)
            msg.attach(mb)

        server = smtplib.SMTP_SSL("smtp.gmail.com",465,context = ssl.create_default_con
        server.login(gmail_account,gmail_password)
        server.send_message(msg)
        os.remove('/home/pi/json_folder/{}.json'.format(td5))
        print("メールの送信が完了しました")
        json_load=[]
        os.remove('/home/pi画像//{}.JPG'.format(td5))
else :
        return
```