卒業論文

語彙学習を行うシステムの開発

関西学院大学理工学部

情報科学科 西谷研究室

27014531 宮下　優理

2018年3月

目 次

第1章　はじめに

仕事において外国語を使うことが不可欠となった場合，私たちは最短最速で外国語の学習をしなければならない．外国語学習は私たちにとって身近なもので，英語は中学校から主要科目として時間をかけて学んできた．しかし，自身の英語力に自信を持っている者は少なく，巷には「英語習得法」などと銘打った情報が溢れている．多くの人々は自分の学習スタイルに不満を感じている．これでは思うように英語力を伸ばすことは難しい．

一般的に英語を仕事で使うためのラインは，TOEIC700 点以上と言われている [1]．これは TOEIC 受験者の上位約 30%に当たる点数である [2]．一方で，2015 年に IP テストを受験した大学 1 年生の平均は 417 点で，大学 4 年生の平均点は 502 点であった [3]．この結果は 4 年間かけて伸びた点数が 85 点であることを示しており，同じペースで学習を進めると仮定すると，大学 4 年生の時点から約 200 点もの点数アップを実現するには 10 年もの月日がかかるということである．最短最速で英語学習を進めるには学習方法を見直し，今までのやり方を変えなければならない．

外国語の習得において学習開始前に不足しているスキルは単語，文法，読解，速読，発音，会話などと人それぞれであると考えられるが，まずは外国語学習の最も基礎となる単語力に着目する．TOEIC700 点に達するために必要な語彙数は 8000 語だと言われている [4]．また，大学 4 年生の平均に近い TOEIC500 点での語彙力は 4000 から 5000 語と言われているため，その差は 3000 から 4000 語である [5]．そこで，本研究では語彙力を身につけるための語彙学習システムの開発を試みる．

はじめに，語彙を「覚える」ことの効率を上げるため，人の「記憶」について調べた．近年では脳の構造からみた記憶のメソッドが多く提唱されている．これらを学ぶことで，内容を脳に定着させ易くしたり，過去に記憶したものを思い出し易くすることができる．また，記憶には様々な種類があり，その発達段階は年齢によって異なっている [6, p.188]．若い頃は意味記憶がよく発達しているため，丸暗記することが得意である．そして，中学生を過ぎるとエピソード記憶が発達し，論理立てて記憶することが得意となる．大人が学

習を行う際は，論理立てて記憶を形成することが必要なのである．そして，これらのメソッドは語彙学習に活かすことができる．

現時点で多くの語彙学習アプリが開発されている．アプリの内容はクイズ形式のもの，自動で暗記度を判定してくれるもの，既知のものはリストから消せるものなど様々であった．実際にこれらを利用してみると，大人が学習する際に大切な論理立てた記憶や連想的な記憶を補助するシステムが用意されていないことがわかった．そこで私の開発するシステムは，語彙に関するメモの自由度を高くすることで，自身の持つ記憶と結び付け易いメモを残せるものとした．システムの主な機能は2つで，事前に用意した語彙リストからチェックテストを行えることと，記憶できていない語彙に対してメモを作成できることである．

第2章　基本的事項

2.1　記憶

2.1.1　一般的な記憶技術

学習を行う際の正しい心構えを知るため，はじめに人の「記憶」について調べた．記憶は学習時間に比例するものではなく，また，人の潜在的な能力のみに依存するものでもない．近年では，記憶に関する脳の仕組みが解明されてきており，記憶力を鍛える様々なメソッドが開発されている．私たちはただがむしゃらに学習しているだけでは，理想的な成果を上げられない．開発されたメソッドを取り入れることで，効率的な学習システムや環境の構築ができるだろう．

アルベルト・オリヴェリオによって書かれた「覚える技術」には，次のようなことが書かれている．まず上手に記憶するための戦略として，

1. 脳を健康状態に保つこと
2. 脳の活動を抑制すること
3. 視覚イメージを持つこと

の 3 つが挙げられている [7, p.130-3]．

1 つ目の脳を健康状態に保つことは，神経組織への酸素補給に依存している．そのため適度な肉体運動を行うことは脳への血流を促進し，記憶力の維持に役立つ．

2 つ目の脳の活動を抑制することは，一見すると記憶を妨げるように思われるが，神経を高ぶらせる内外の刺激をうまく断ち切るということである．これを実践することができれば，集中力が高まる．チェスの名人たちは，これを用いてストレスを解消したり注意力を養ったりしている．

3つ目の視覚イメージとは，メモ書きや図表を使ってキーとなる概念を見極め，それを視覚化して心の中でイメージして描くというものである．このイメージは慣れてくると記憶を心に呼び戻したり，甦らせたりする有力なヒントとなる．

次に，視覚イメージを利用する「場所」のメソッドは記憶力を強化するとされている [7, p.116]．場所の記憶は，あらゆる記憶の中で驚異的に残る．古代ギリシャ人や古代ローマの雄弁家たちも利用してきたこの方法は，ひとつの場所に当てはめた記憶の道筋をたどっているのである．つまり，記憶したいものと関連する場所をセットにして覚えることが，記憶を助けることとなる．

さらに，覚えやすいものと覚えにくいものについてもいくつか記述されている．

- 文字列は長いほど検索が容易である [7, p.145]．
- 記憶を呼び覚ますこととは納得できる，連想できる記憶を構成し，再構成する作業を始めることである [7, p.148]．
- 強い印象の残る経験でなければ記憶は定着しない [7, p.65]．
- 類似のものは互いにその記憶を消し去ってしまう傾向がある [7, p.66]．
- 類似のものでも間隔をあけると干渉を防げる [7, p.71]．
- 短期記憶は一度に 6 個か 7 個の情報を保存する能力があるため，7 個以下の情報となるようグループ分けをする [7, p.63]．

最後に，感情と記憶は相関が強いことも言及されている [7, pp.231-5]．不安や恐怖を感じている時，周知の事実であってもなかなか思い出せないようなことがある．また，興奮状態にありアドレナリンが出ている時には，記憶の定着が促される．

アドレナリンの分泌が引き起こす記憶の定着について，次のような言い伝えがある．

> 中世において記述記憶，つまり紙が高価で利用できなかった市中においては子供の記憶が利用されていた．何かのいさかいがあるとそれに関する情報，事情，経緯をじっくりと繰り返し 7，8 歳の子供に記憶させ，大体覚えたところで川へ投げ込む．そうすると係争という最も記述的な記憶が，感情的なそして印象が薄まることのない経験記憶として定着させることができた

というのである [8, pp.5-6]．

また，池谷裕二によって書かれた「記憶を強くする」には，

目の記憶よりも耳の記憶のほうが心に残る

とされている [6, p.200]．歌の歌詞がそのまま見て覚えるよりもメロディーと一緒に覚えるほうが記憶に残りやすいことは，誰もが経験していることだろう．

2.1.2 記憶技術を語彙学習に生かす

上記に記した記憶のポイントを語彙学習に当てはめると，

- 連想できる構成をすること
- グループ分けを行うこと
- 印象を強くすること
- 耳の記憶を活用すること

の 4 点を特に活用することができる．システムを利用してメモを作成する際，以下のことを意識すると優れたメモを作成できるだろう．

連想できる構成を作る糸口として，2 つあげる．1 つは「語幹」である．動詞，名詞，形容詞，副詞で同じ語幹を持つものは数多く存在する．そのうち 1 つを記憶することで，他の単語は連想的に思い出すことができる．もう 1 つは「語源」だ．接頭語，語根，接尾語には語源がある．これらが組み合わさることで多くの単語が形成されているため，語源を覚えることは単語を連想的に思い出すことにつながる．

グループ分けは，類義語ごとに行うことを提案する．例えば，complete の類義語は end, finish, be over, terminate などであると言え，5 個ほどの適した数の要素数を持つグループが形成できる．

印象を強くするためには，語彙と例文はひとまとまりにする．より強い印象ため，記憶する時の状況，自身のエピソードを活用した例文を作成する．また，例文にすることで文字列を長くなり，思い出し易くすることができる．

耳からも記憶にアプローチできるよう，本システムには単語の発音を再生できる機能を取り入れている．正しい発音を耳から入れることで，目で見るだけよりも強い刺激で記憶に残らせ易くすることができる．

2.2　既存の英単語アプリと開発システムのコンセプト

2.2.1　既存の英単語アプリ

コンピュータやインターネットを利用した多くの英単語アプリやメモ帳アプリが提供されている．評価の高かった [9] いくつかのアプリの機能や弱点について考察する．

- mikan[10]
  - 機能
    - ４択の英単語クイズ．
    - 間違った単語は苦手な単語として記録される．
    - 知ってる or 知らないを選択し，選択すると意味が表示される．
    - 間違った場合再表示され，そのターンの単語が全て知ってるになるまで続く．
    - 総復習では間違った回数によって「完璧に覚えた」,「ほぼ覚えた」,「うる覚え」と分類される．
  - 弱点
    - 間違ったものがすぐに立て続けに出てくるので，英単語の記憶というよりその場しのぎの暗記になってしまう．
    - その間違えた回数によって定着のレベルが判断されるので，信憑性が低い．
- 究極英単語！TOEIC 800 点突破編 [11]
  - 機能
    - タイトルレベルにあった単語の４択クイズ．
    - 回答せずに答えを表示，わかった，わからないを選択することもできる．
    - 復習専用ページはなく，間違えた単語は自動でクイズ中に何度も登場する．
    - 完成，習得，習得中の指標で，習得レベルの確認ができる．
  - 弱点
    - 概ね mikan と同様．
- 最後の英単語学習！マジタン (有料)[12]

- 機能
  - 語彙数カウント機能で自分の学習達成度を把握できる
  - スクリーニング機能で既に知っている単語は省ける．
  - 記憶度変化機能で一度覚えた単語が一定時間経過後に再び登場する．
- 弱点
  - スクリーニング機能において，単語の左端に「既知」と表示されるだけのため，単語は目につき，あまりスクリーニングとしての効果を発揮できていない．

- Quizlet[13]
  - 機能
    - 自身で単語リストの作成を行う．
    - 英→日，日→英それぞれの選択肢クイズ，書き込み式クイズとテストの形式が豊富である．
  - 弱点
    - web アプリでは単語帳を作成するのに時間がかかる．

2.2.2 開発システムの概要

前節で調べたアプリの問題点をまとめると次の通りとなる．まず，基本的な操作のメタファとしては単語帳，あるいは紙媒体の単語本がある．これをネットあるいはアプリに焼き直して提供している．その内容については 2 種類あり，提供された内容をチェックするか，あるいはその内容を生成していくかに分類される．

覚える技術で調べた通り，英語学習においては自分の経験に照らして類推，連想によって学習していくことが効率を上げるコツとされている．また，音声などの記憶に残りやすい情報をもとに検索，整理することが望まれる．

したがって，自分にあった語彙や，その内容を作り変えることの困難さが学習を妨げていることが考えられる．また，語幹や類語，語源，例文を自由に編集することができない．さらに，英英のシステムにしようとすることが難しい．もっとも共通する特徴は web アプリのため，GUI(Graphical User Interface) での操作が当たり前なところである．

一方，開発しようとするシステムの最も特徴的なことはCUI(Character User Interface)での提供である．システムの開発者はプログラマーなのでCUIでの作業環境が当たり前である．プログラマーは，コマンドラインを使ったシステムの操作に習熟している．そこで，学習ソフトの最初のユーザーとしてプログラマーを想定し，このアプリもCUIで提供することを考える．

さらに単語帳の編集には，プログラマになじみのエディターであるemacsでの提供を考えている．emacsとテキストベースのcsvファイルの組み合わせによって，覚えたい単語のリストの作成や追加，単語ごとに必要事項のメモがなじみの環境で操作できる環境の提供を目論んでいる．また，学習の進歩が視覚化できるように，学習記録の確認のできる機能を提供する．また，OSXで提供されている`say`というコマンドラインアプリケーションを使って音声での出力，確認ができる機能を提供する．

2.3 使用したライブラリとデータフォーマット

2.3.1 Gemパッケージ

Rubyで使われるライブラリやアプリケーションはGemと呼ばれる形式のパッケージにすることができる．多くのライブラリがGem形式でパッケージされ公開されており，これらはRubyGemsと呼ばれるパッケージ管理ツールを使ってダウンロードを行なったりインストールすることができる．これらのパッケージのことをGemやGemパッケージと呼ぶ．

私が開発するシステムもinstallやdocumentの保守性を考えて標準的なgem packageで提供できるように開発を進めた．

2.3.2 Bundler

Bundlerとはアプリケーション単位で依存するgemパッケージを管理するためのツールである．gemパッケージを作るには，gemパッケージの雛形を生成するコマンドを利用してgemパッケージの土台作りをすることが一般的である．最近はこの「Bundler」が主流となっている [14, ch.13]．

2.3.3　コマンドラインアプリ構築ライブラリ

コマンドラインアプリ構築ライブラリとして ruby においては optparse が標準で添付されている．しかし，Build Awesome Command-Line Applications in Ruby2[15] で説明されている通り，オプションの書き方として自然言語に近い形での表記がユーザーに広く使われるようになっている．そのため，この表記に対応しやすい Thor をライブラリとして利用する．

2.3.4　データ構造

実働する application をつくるならば高速化，大容量化を見越して MySQL などの free の relational data base(RDB) を使うべきである．しかし，本研究においては vocaburary building のシステムの実験的な開発を目的としているので，それほど大容量化に対応する必要がないと考え，debug や視認が容易なテキストベースでのデータ構造を採用した．

基本的には，comma-separated values(csv) 形式を採用している．ただし，ファイル名に空白を使うことが難しいいので，idiom の表記には空白を underbar に変えた snake 形式での記述を採用している．

2.3.5　音声出力対応

英語は，中学校以来習っているため，文字情報によってある程度発音が予測できる．しかし，フランス語や，スペイン語，ドイツ語，中国語のように日本人にとって発音が難しいとされる言語を耳から学習するためにも音声情報は重要となる．また，英語においても耳からの音声情報を key として記憶に定着させることも期待できる．本システムでは音声での出力が可能となるようなオプションを用意する．

OSX の terminal command の中には，

```
say 'hello'
```

などとして使える読み上げソフトが組み込まれている．これを開発中のシステムに組み込むことによって音声として聞き取ることが可能である．

言語の音声の選択については次のような option が用意されている．

say -v 'Alex' Hello

によって Alex と名付けられた voice により読み上げが行われる． Alex は iOS 8 以降では，VoiceOver，画面の読み上げ，および選択項目の読み上げの英語の読み上げ音声として用意されている標準的な音声である．

say -v '?'

で say が用意している音声のすべてのリストが表示される．そこでは，その音声が喋る，言語と国を表す記号が付されている．また，その voice の特徴がもっとも表されるセンテンスがある．この一部を示すと次の通りである．

```
Alex                en_US   # Most people recognize me by my voice.
Alice               it_IT   # Salve, mi chiamo Alice e sono una voce italiana.
Bad News            en_US   # The light you see at the end of the tunnel
                              is the headlamp of a fast approaching train.
Bells               en_US   # Time flies when you are having fun.
Daniel              en_GB   # Hello, my name is Daniel. I am
                              a British-English voice.
```

この option を使って，次のように音声選択の名前を明示することで，

```
> say -v 'Kyoko' こんにちは、私の名前は Kyoko です。日本語の音声をお届けします。
> say -v 'Alice' Salve, mi chiamo Alice e sono una voce italiana.
```

とすれば，日本語，イタリア語など，他の言語の読み上げが可能である．

本研究においては，語彙のメモの作成時に英単語の音声読み上げ機能を組み込んでいる．

第3章　開発システムの解説

yuris_vocabuil では，覚えたい単語のリストの作成や追加，単語ごとに必要事項のメモ，学習記録の確認のできるシステムである．vocabuil とは vocaburary building を略称している．

3.1　使用法

3.1.1　インストール

rake install yuris_vocabuil

yuris_vocabuil をインストールする．

rake make_dir

yuris_vocabuil をインストールする際に必要なファイル，フォルダを作成する．

本システムは，ターミナルのどのディレクトリにおいても実行することができる．

3.1.2　directory 構成および初期設定

ホームディレクトリに.yuri_vocabuil_USERNAME というフォルダが生成される．その中の階層は，

```
└── lib
    ├── backup
    │   └── vocabulary_list_backup.csv
    ├── card_dir
    └── vocabulary_list.csv
```

となっている．

card_dir のフォルダは，初期設定では空であり，edit で作成するテキストファイルが入る．vocabulary_list.csv も初期設定では空であり，check で用いる単語リストである．edit でテキストファイルを作成した単語は自動的にこのリストに書き込まれる．また，vocabulary_list.csv の左端の列に直接単語を書き込み，リストを作成しても良い．

3.1.3 ルーチン操作及び実行時の様子

ターミナルで以下のように入力すると，動作させることができる．

yuris_vocabuil edit word

word には，編集したい，調べたい語彙を入れる．word が熟語の場合には，スペースでなくアンダーバーを挿入する．word 中にアポストロフィが入る場合には，アポストロフィの直前にバックスラッシュを挿入する．

これを実行すると，word が発音され，オンラインの百科事典サイトである weblio の該当ページが開き，ターミナルでは word.txt のファイルが emacs で開く．サイト等を見ながら，word.txt に自由なメモを残すことができる．作業が終了したら，ファイルを保存し emacs を閉じる．また，vocabulary_list に word が存在しない場合，リストに追加される．

yuris_vocabuil check

vocabulary_list からランダムに選ばれた語彙が選択される，テストのようなものである．表示される「d(one), e(dit), or q(uit) を入力 [d,e,q]: 」に従って，標準入力を行う．ここで入力したものは，時間とともに記録される．理解のできているものは d，学習が必要なものは e を選択する．e と入力すると edit へ移行し学習を行える．標準入力を行うと，新たに語彙がランダムに選択され，表示される．check を終わる時には q を入力する．

yuris_vocabuil achievement

check でテストした数を日付ごとに表示する． 学習進捗状況の確認ができる．以下のように表示される．

Hello, Ruby! achievement
2018/01/10:****
2018/01/17:**********
2018/01/18:
2018/01/19:
2018/01/24:**********
2018/01/25:**********
2018/01/26:**********
2018/01/29:**********
2018/01/30:**********
2018/01/31:****
2018/02/06:
2018/02/07:
2018/02/08:
2018/02/09:***

yuris_vocabuil delete word

word には次の動作を行いたい語彙を入れる．誤った語彙のファイル作成とリストへの追加を行った際に，それらを削除することができる．

yuris_vocabuil version

システムの version を表示する．

3.2　必要な機能と実装

3.2.1　edit

1. 標準入力した単語，熟語は card に入る．card では，card 名.txt のファイルを作成する熟語の単語間のスペースの箇所にはアンダーバーを使用している．また，card は web サイトに接続する時や emacs でテキストファイルを開く時，say コマンドに用いる時のために，gsub によって card_web や card_emacs，card_pro などと微調整している．

2. say コマンドを実行し，語彙の発音を行う．

3. system(”open https://ejje.weblio.jp/content/#{card_web}”) を実行し，該当サイトを開く．

4. 既に vocabulary_list に該当するテキストファイルが存在する場合，それを emacs で開き，存在しない場合はそのテキストファイルを新規作成し，emacs で開く．

5. 語彙が vocabulary_list に存在しない場合は追加する．lines で check で用いる csv ファイルを 1 行ずつ読み込み．各行の 1 列目に card と一致する単語があるかを探索し，あった場合は for 文から抜ける．カウントが行数と一致した場合と vocabulary_list に何も入ってなかった場合の 1 で会った時，vocabulary_list に語彙が存在しないことを示すため，リストに追加し，”new history added.” と表示する．

6. edit コマンドを用いて学習した記録を残すため，status を”e” として add_date_stamp 関数により，csv ファイルに日時と status を記録する．

```
desc '-e', 'edit word card'
map '-e' => 'edit'
def edit(*argv)
  puts "open specific word card of"
  card = argv[0]
  now = Time.new
  puts ("#{card}")

  card_web = card.gsub('_','+')
  card_emacs = card.gsub(/\'/){'\\\''}
  card_web = card_web.gsub('\'', '%27')
  card_pro = card.gsub('_',' ')
  card_pro = card_pro.gsub("\'",'')

  system "say -v \'Alex\' #{card_pro}"

  @mk_card_dir_location =
  File.join(@local_help_dir,"lib/card_dir/#{card_emacs}.txt")

  system("open https://ejje.weblio.jp/content/#{card_web}")
  if File.exist?(@mk_card_dir_location)
    system("emacs #{@mk_card_dir_location}")
  else
    File.open(@mk_card_dir_location, "w")do |f|
      system("emacs #{@mk_card_dir_location}")
    end
  end
```

```
    # add card to vocabulary_list.csv
    lines = CSV.readlines(@vocabulary_list)
    p max_n = lines.size
    sel = 1
    File.open(@vocabulary_list,"a") do |file|
      for sel in 1..max_n-1 do
        if card == lines[sel][0]
          sel = sel-1
          break
        end
      end

      if sel == max_n-1 || max_n == 1
        file.puts "#{card}"
        puts "new history added."
      end
    end
    status='e'
    add_date_stamp(card,status)
  end
```

3.2.2 check

1. lines で check で用いる csv ファイルを 1 行ずつ読み込む．max_n に csv ファイルの行数が入る．

2. 以下をループで回す．sel に 0 から max_n-1 の数字からランダムに選ばれたものが入る．line に sel 行目の単語が入る． 青文字で line を表示する．history_size にはその単語の行の列数，つまりその単語の表示が何度目があるかを示す数字が入り，表示される．

3. 赤文字で'd(one), e(dit), or q(uit) を入力 [d,e,q]:' と表示される．

    1. 'e' と入力すると，その単語が引数となり edit 関数へ飛ぶ．

2. 'd' と入力すると，'good job!' と表示され，status に d が入る．line, status の 2 つを引数として add_date_stamp 関数に飛ぶ.
3. 'q' と入力すると，ループから抜け出す.

```
desc '-c', 'check word'
map '-c' => 'check'
def check
  puts "check random word"
  lines = CSV.readlines(@vocabulary_list)
  p max_n = lines.size
  loop do
    p sel = rand(max_n)
    line = lines[sel][0].chomp
    puts line.blue

    history_size = lines[sel].size
    puts "#{history_size}回目"

    print 'd(one), e(dit), or q(uit) を入力 [d,e,q]: '.red
    input = STDIN.gets.chomp
    p input

    case input
    when 'e'
      edit(line)
    when 'd'
      puts 'good job!'
      status='d'
      add_date_stamp(line,status)
    when 'q'
      exit
    end
  end
end
```

3.2.3 delete

1. 標準入力した単語，熟語は card に入る．
2. 'Are you sure?(y/n)' と表示する．
3. y と入力すると，system("rm #{CARD_DIR}/#{card}.txt") によりテキストファイルを削除し，'delete card 名.txt' と表示する．
4. n と入力すると，'canceled' と表示されるのみである．

```
desc '-d', 'delete word card'
map '-d' => 'delete'
def delete(*argv)
  card = argv[0]

  puts 'Are you sure?(y/n)'
  input = STDIN.gets.chomp
  p input
  card_rm=card.gsub(/\'/){'\\\''}

  # mk backup at a moment
  File.open(@vocabulary_list_backup,"w") do |file|
    File.open(@vocabulary_list,"r") do |read|
      read.each_line do |l|
        file << l
      end
    end
  end
```

```
    case input
    when 'y'
      system("rm #{CARD_DIR}/#{card_rm}.txt")
      puts ("delete #{card}")
      File.open(@vocabulary_list,"w") do |file|
        File.open(@vocabulary_list_backup,"r") do |read|
          read.each_line do |l|
            if l.include?(card)
              str=l.match(/#{card}\w+/)
              str2=l.match(/\w+#{card}/)
              if str == nil && str2 == nil
                content = ""
              else
                content = l
              end
              str=nil
              str2=nil
            else
              content = l
            end

            file << content
          end
        end
      end
    when 'n'
      puts 'canceled'
    end
  end
```

3.2.4 achievement

1. lines で vocabulary_list.csv を 1 行ずつ読み込む.
2. total に csv ファイルの行数が入る.

3. history を空にする．history は key に日付，value にその日付が入っている箇所の数が入るハッシュである．
4. lines を 1 行ずつ順に見ていく．
5. lines[i].size > 1 つまり date_stamp が記載されているなら，記載されているものを左から順に見ていく．
6. day[1] に date_stamp の日付部分を格納する．
7. history[day[1]]!=false つまり day[1] に格納されている日付の登場が初めてならば 1 を格納し，そうでなければ history[day[1]] の数に 1 加える．
8. day は nil に戻しておく．
9. 3 箇所ごとに 1 個の米印を表示し，過去の日付の進捗具合がわかる．

```
desc '-a', 'show achievement'
map '-a' => 'achieve'
def achieve
  puts "achievement "
  lines = CSV.readlines(@vocabulary_list)
  total = lines.size
  history ={}

  for i in 1..total-1 do
    if lines[i].size > 1
      for j in 1..lines[i].size-1
        if day = lines[i][j].match(/(\d{4}\/\d{2}\/\d{2}) \d{2}:\d{2}/)
          if day[1] != nil
            if  history[day[1]] == false
              history[day[1]] = 1
            elsif history[day[1]] != false
              history[day[1]] = history[day[1]].to_i + 1
            end
          end
          day = nil
        end
      end
    end
  end

  history.sort.map do |key,value|
    print "#{key}:"
    for i in 1..value/3 do
      print "*"
    end
    print "\n"
  end
end
```

第4章　総括

本研究では学習を進められる程度のシステムを開発することができた．以下に開発内容を簡潔に示す．

1. 学びたい語彙を入力すると，その語彙名のついたテキストファイルとオンライン百科事典サイトが開き，その語彙に関する柔軟な学習が行える．
2. 登録された語彙リストからランダムに語彙の意味を理解できているか試すテストが行える．
3. テストにて学習した記録は残り，自身の学習進捗状況を確認することができる．

今後の展望としては，スマートフォンで動作可能なアプリケーションと連携させることを考えている．本研究で開発したシステムで学習を行ったのち，そのメモと語彙リストをアプリケーションに同期させ，アプリケーションでの復習を可能にする．より身近に，隙間時間を有効活用して学習を行えるようになる．

また，音声機能のさらなる充実も図っていきたい．本研究では英語の語彙の発音を聞くことのできるシステムとなった．今後はあらかじめ言語を選択することで，他の言語にも対応させたい．また，語彙のみでなくメモに入力した例文や関連する語彙も自由に発音させられるものとしたい．耳からの情報を積極的に取り入れ，深い記憶を形成していく．

謝辞

本研究を行うにあたり，終始多大なるご指導，御鞭撻をいただいた西谷滋人教授に対し，深く御礼申し上げます．また，本研究の進行に伴い，様々な助力，知識の供給を頂きました西谷研究室の同輩，先輩方に心から感謝の意を示します．本当にありがとうございました．

参考文献

[1] ManpowerGroup,「採用条件にある『ビジネス英語ができる方』とは、どの程度のスキル?」, https://www.manpowergroup.jp/column/career/151225_01.html, (accessed 2018/2/5).

[2] TOEIC,「公式スコア・スコア分布詳細 (第 226 回)」, http://www.iibc-global.org/toeic/official_data/lrdata_avelist/226.html#anchor02, (accessed 2018/2/5).

[3] TOEIC,「Program DATA & ANALYSIS 2016」, http://www.iibc-global.org/library/toeic_data/toeic/pdf/DAA.pdf, (accessed 2018/2/5).

[4] 99(ナイナイ) ブログ,「TOEIC で必要な語彙数は?あなたの今の語彙数はどれくらい?」, http://www.99institute.com/blog/ 2017/ 04/ toeic で必要な語彙数は?あなたの今の語彙数はどれ.html, (accessed 2018/2/5).

[5] 英語の歩き方,「TOEIC500 点突破に必要な語彙力」, https://english-hack.net/memorize-english-words-less-than-500-toeic/, (accessed 2018/2/5).

[6] 池谷祐二,「記憶力を強くするー最新脳科学が語る記憶の仕組みと鍛え方」, (ブルーバックス, 2001).

[7] アルベルト・オリヴェリオ,「覚える技術」, (翔泳社, 2002).

[8] ジェームズ・L・マッガウ,「記憶と情動の脳科学」, (講談社, 2006).

[9] Appliv,「英単語・ボキャブラリーを覚える アプリランキング」, https://app-liv.jp/education/languages/1049, (accessed 2018/2/5).

[10] Appliv, 「英単語アプリ mikan」, https://app-liv.jp/920856839/, (accessed 2018/2/14).

[11] Appliv, 「英単語アプリ究極英単語 TOEIC800 突破編」, https://app-liv.jp/868628106/, (accessed 2018/2/14).

[12] Appliv, 「英単語アプリ最後の英単語学習!マジタン」, https://app-liv.jp/872461032/, (accessed 2018/2/14).

[13] Quizlet, 「Quizlet」, https://quizlet.com/ja, (accessed 2018/2/14).

[14] Ruby サポーターズ, 「パーフェクト Ruby」, (技術評論社, 2014).

[15] David Bryant Copeland, "Build Awesome Command-Line Applications in Ruby 2", (Pragmatic Bookshelf, 2013).