卒業論文

command line による editor 操作の習熟プログラム

関西学院大学理工学部

情報科学科 西谷研究室

27014533

2018 年 3 月

目 次

表 目 次

図 目 次

第1章　はじめに

1.1　研究の目的

editor_learner の開発の大きな目的は editor(Emacs) 操作，CUI 操作 (キーバインドなど)，Ruby 言語の習熟とタイピング速度の向上である．editor 上で動かすためファイルの開閉，保存，画面分割といった基本操作を習熟することができ，Ruby 言語のプログラムを写経することで Ruby 言語の習熟へと繋げる．更にコードを打つことで正しい運指を身につけタイピング速度の向上も図っている．コードを打つ際にキーバインドを利用することで GUI ではなく CUI 操作にも適応していく．これら全てはプログラマにとって作業を効率化させるだけでなく，プログラマとしての質の向上につながる．

1.2　研究の動機

初めはタッチタイピングを習得した経験を活かして，西谷によって開発された shunkun-type(ターミナル上で実行するタイピングソフト) の再開発をテーマにしていたが，これ以上タイピングに特化したソフトを開発しても同じようなものが Web 上に大量に転がっており，そのようなものをいくつも開発しても意味がないと考えた．そこで西谷研究室ではタイピング，Ruby 言語，Emacs による editor 操作，CUI 操作の習熟が作業効率に非常に大きな影響を与えるので習熟を勧めている．そこで西谷研究室で使用されている editor である Emacs 操作，Ruby 言語の学習，タイピング速度，正確性の向上，CUI 操作．これらの習熟を目的としたソフトを開発しようと考えた．

第2章　基本的事項

2.1　Emacs

本研究において使用するeditorはEmacsである．ツールはプログラマ自身の手の延長である．これは他のどのようなソフトウェアツールよりもEditorに対して当てはまる．テキストはプログラミングにおける最も基本的な生素材なので，できる限り簡単に操作できる必要があります．[1] そこで西谷研究室で勧められているEmacsの機能については以下の通りである，

1. 設定可能である．　フォント，色，ウィンドウサイズ，キーバインドを含めた全ての外見が好みに応じて設定できるようになっていること．通常の操作がキーストロークだけで行えると，手をキーボードから離す必要がなくなり，結果的にマウスやメニュー駆動型のコマンドよりも効率的に操作できるようになります
2. 拡張性がある．　新しいプログラミング言語が出てきただけで，使い物にならなくなるようなエディタではなく，どんな新しい言語やテキスト形式が出てきたとしても，その言語の意味合いを「教え込む」ことが可能です
3. プログラム可能であること．　込み入った複数の手順を実行できるよう，Editorはプログラム可能であることが必須である.

これらの機能は本来エディタが持つべき基本的な機能である．これらに加えてEmacsは，

1. 構文のハイライト　Rubyの構文にハイライトを入れたい場合はファイル名の後に.rbと入れることでRubyモードに切り替わり構文にハイライトを入れることが可能になる．
2. 自動インデント．　テキストを編集する際，改行時に自動的にスペースやタブなどを入力しインデント調整を行ってくれる．

などのプログラミング言語に特化した特徴を備えています．強力なeditorを習熟することは生産性を高めることに他ならない．カーソルの移動にしても，1回のキー入力で単語単位，行単位，ブロック単位，関数単位でカーソルを移動させることができれば，一文字ずつ，あるいは一行ずつ繰り返してキー入力を行う場合とは効率が大きく変わってきます．Emacsはこれらの全ての機能を孕んでいてeditorとして非常に優秀である．よって本研究はEmacsをベースとして研究を進める．

2.2 Ruby

Rubyの基本的な説明は以下の通り，

Rubyはまつもとゆきひろにより開発されたオブジェクト指向スクリプト言語であり，スクリプト言語が用いられてきた領域でのオブジェクト指向プログラミングを実現する言語である．[1]

本研究はRuby言語を使用しています．大きな理由としては * 構文の自由度が高く，記述量が少なくて済む． * 強力な標準ライブラリが備えられている．

Rubyは変数の型付けがないため，記述量を少なく済ませることができ，”gem”という形式で公開されているライブラリが豊富かつ強力なので本研究はRuby言語を使用しました．

2.3 RubyGems

Rubygemの基本的な説明は以下の通り，

RubyGemsは，Ruby言語用のパッケージ管理システムであり，Rubyのプログラムと（"gem"と呼ばれる）ライブラリの配布用標準フォーマットを提供している．gemを容易に管理でき，gemを配布するサーバの機能も持つ．[2]

本研究ではRubyGemsのgemを利用してファイル操作やパスの受け取りなどを行う．

2.4 Keybind

Keybind の基本的な説明は以下の通り，

押下するキー (単独キーまたは複数キーの組み合わせ) と，実行される機能との対応関係のことである．また，キーを押下したときに実行させる機能を割り当てる行為のことである．[3]

以下 control を押しながらを c-と記述する．本研究における Keybind の習熟は CUI 操作の習熟に酷似している．カーソル移動においても GUI ベースでマウスを使い行の先頭をクリックするより，CUI により c-a を押すことで即座に行の先頭にカーソルを持っていくことができる．習熟するのであれば，どちらの方が早いかは一目瞭然である．本研究は Keybind の習熟による CUI 操作の適応にも重点を置いている．

2.5 CUI(Character User Interface)

CUI は，

キーボード等からの文字列を入力とし，文字列が表示されるウィンドウや古くはラインプリンタで印字される文字などを出力とする，ユーザインタフェースの様式で，GUI(Graphical User Interfa
の対義語として使われる．[4]

CUI と GUI にはそれぞれ大きな違いがある．GUI の利点は以下の通り，

- 文字だけでなくアイコンなどの絵も表示できる．
- 対象物が明確な点や，マウスで比較的簡単に操作できる．
- 即座に操作結果が反映される．

CUI の利点は以下の通り，

- コマンドを憶えていれば複雑な処理が簡単に行える．
- キーボードから手を離すことなく作業の高速化が行える．

今回 GUI ではなく CUI 操作の習熟を目的にした理由は，

- コマンドを憶えることで作業効率が上がる．
- editor 操作の習熟も孕んでいるから．

カーソル移動においても GUI ではなく CUI 操作により，ワンコマンドで動かした方が効率的である．上記の理由から，GUI ではなく CUI 操作の習熟を目的としている．

2.6 使用した gem ファイル

2.6.1 diff-lcs

diff-lcs は，二つのファイルの差分を求めて出力してくれる．テキストの差分を取得するメソッドは,Diff::LCS.sdiff と Diff::LCS.diff の 2 つがある. 複数行の文字列を比較した場合の 2 つのメソッドの違いは以下のとおり.

1. Diff::LCS.sdiff 比較結果を 1 文字ずつ表示する
2. Diff::LCS.diff 比較した結果，違いがあった行について，違いがあった箇所のみ表示する．

今回使用したのは後者 (Diff:LCS.diff) である．理由は間違った部分だけを表示した方が見やすいと考えたからである．

2.6.2 Thor

Thor は，コマンドラインツールの作成を支援するライブラリです．git や bundler のようなサブコマンドツールを簡単に作成することができます．Thor の使用で容易にサブコマンドを直感的に覚えやすくした．

2.6.3 Minitest

Minitest はテストを自動化するためのテスト用のフレームワークである．Ruby にはいくつかのテスティングフレームワークがありますが，Minitest というフレームワークを利用した理由は以下の通りです．

1. Ruby をインストールすると一緒にインストールされるため，特別なセットアップが不要.
2. 学習コストが比較的低い.
3. Rails のデフォルトのテスティングフレームワークなので，Rails を開発するときにも知識を活かしやすい.

2.6.4 FileUtils

再帰的な削除などの基本的なファイル操作を行うためのライブラリ

2.6.5 open3

プログラムを実行し，そのプロセスの標準出力，標準入力，標準エラー出力にパイプをつなぐためのものである.

2.7 Bundler

Bundler はアプリケーション谷で依存する gem パッケージを管理するためのツールです．1 つのシステム上で複数のアプリケーションを開発する場合や，デプロイ時にアプリケーションに紐付けて gem パッケージを管理したい場合に利用される.

2.7.1 Rubocop

Rubocop は Ruby のソースコード解析ツールである．Ruby スタイルガイドや他のスタイルガイドに準拠しているかどうかを自動チェックしてくれるソフトウェアです．自分が打ち込んだ問題文となるソースコードのチェックに使用しました．[]

2.8 editor_learnerの概要

2.9 Installation

2.9.1 githubによるinstall

githubによるインストール方法は以下の通りである．1. ”https://github.com/souki1103/editor_learner
へアクセス 1. Clone or downloadを押下，SSHのURLをコピー 1. コマンドラインにて
git clone [コピーしたURL] を行う

上記の手順で開発したファイルがそのまま自分のディレクトリにインストールされる．

2.9.2 gemによるinstall

gemによるインストール方法は以下の通りである．1. コマンドラインにてgem install ed-
itor_learnerと入力，実行 1. ファイルがホームディレクトの.rbenv/versions/2.4.0/lib/ruby/gems/2.4.0/ge
にeditor_learnerが収納される

これでeditor_learnerとコマンドラインで入力することで実行可能となる．

2.10 uninstall

2.10.1 githubからinstallした場合のuninstall方法

gituhubからinstallした場合のuninstall方法は以下の通りである．

1. ホームディレクトで
    2. rm -rf editor_learnerを入力
2. ホームディレクトリからeditor_learnerが削除されていることを確認する．

以上がuninstall方法である．

2.10.2 gem から install した場合の uninstall 方法

gem から install した場合の uninstall 方法は以下の通りである．

1. ターミナル上のコマンドラインで

    2. gem uninstall editor_learner を入力

2. ホームディレクトの.rbenv/versions/2.4.0/lib/ruby/gems/2.4.0/gems に editor_learner が削除されていることを確認する．

以上が uninstall 方法である．

2.11 動作環境

Ruby の version が 2.4.0 以上でなければ，動作しない．gem で install した際に gem で install したものが格納されているパスを正常に受け取ることができないためである．

2.12 初期設定

特別な初期設定はほとんどないが起動方法は以下の通りである，

1. コマンドライン上にて editor_learner を入力する．

    2. editor_learner を起動することでホームディレクトリに editor_learner/workshop と呼ばれるファイルが作成される．workshop は作業場という意味である．
    3. workshop の中に question.rb と answer.rb，random_h.rb と ruby_1$_{\text{ruby_6 が作成され，ruby_1}}$ruby_6 の中に 1.rb~3.rb が作成されていることを確認する．[画像添付]

2. 起動すると以下のようなサブコマンドの書かれた画面が表示されることを確認する．

    Commands: editor_lerner delete [number~number] editor_learner help [COMMAND] editor_learner random_check editor_leraner sequential_check [lesson_number] [1~3numbers]

3. editor_learner の後にサブコマンドと必要に応じた引数を入力すると動作する．それぞれのサブコマンドの更に詳しい説明は以下の通りである．

2.13 delete

editor_learner を起動することで初期設定で述べたようにホームディレクトリに editor_learner/workshop が作成される．delete は workshop に作成された ruby_1~ruby_6 を削除するために作成されたものである．sequential_check で 1 度プログラムを作成してしまうと再度実行すると It have been finished!と表示されてしまうので，削除するコマンドを作成しました．コマンド例は以下の通りである．

コマンド例

1. editor_learner delete 1 3

上記のように入力することで 1 ～ 3 までのファイルが削除される．サブコマンドの後の引数は 2 つの数字 (char 型) であり，削除するファイルの範囲を入力する．

2.14 random_h.rb と sequential_h.rb

random_h.rb と sequential_h.rb が初期設定で作成され，editor_learner を起動することで自動的に作成され，random_check と sequential_check を行う際に最初に開くファイルとなる．random_check 用と sequential_check 用に二つのファイルがある．random_check 用のファイルは以下の通りである．

random_h.rb

```
# to open   question.rb
#  c-x 2: split window vertically
#  c-x c-f: find file and input question.rb
# open answer.rb as above
#  c-x 3: split window horizontally
#  c-x c-f: find file and input answer.rb
# move the other window
#  c-x o: other windw
# then edit answer.rb as question.rb
```

```
# c-a: move ahead
# c-d: delete character
# c-x c-s: save file
# c-x c-c: quit edit
# c-k: kill the line
# c-y: paste of killed line
```

上から順に説明すると， 1. question.rb を開くために c-x 2 で画面を 2 分割にする． 1. c-x c-f で question.rb を探して開く． 1. 次に answer.rb を開くために画面を 3 分割する 1. 同様に c-x c-f で answer.rb を探して開く． 1. c-x o で answer.rb を編集するためにポインタを移動させる． 1. question.rb に書かれているコードを answer.rb に写す．

これらの手順が random_h.rb に記述されている．

次に sequential_h.rb

```
#to open q.rb
# c-x 2: find file and input q.rb
# c-x c-f: find file and input q.rb
# open 1~3.rb as above
# c-x 3: split find file and input 1~3.rb
# move the other window
# c-x o: other window
# then edit 1~3.rb q.rb

# c-a:move ahead
# c-d: delete character
# c-x c-s: save file
# c-x c-c: quit edit
# c-k: kill the line
# c-y: paste of killed line
```

書かれている内容自体は random_h.rb とほとんど差異がないが，開くファイルの名前が

違うため別のファイルとして作成された．この手順に沿って作業することになる．下に書かれているのは主要キーバインドであり，必要に応じて見て，使用する形となっている．

2.15 random_checkの使用方法

random_check の動作開始から終了は以下の通りである．

1. コマンドライン上にて editor_learne random_check を入力
2. 新しいターミナル (ホームディレクトリ/editor_learner/workshop から始まる) が開かれる．
3. random_h.rb を開いて random_h.rb に沿って question.rb に書かれているコードを answer.rb に写す．
4. 前のターミナルに戻り，コマンドラインに”check”と入力することで正誤判定を行ってくれる．
5. 間違っていれば diff-lcs により間違った箇所が表示される．
6. 正しければ新しいターミナルが開かれてから終了までの時間と It have been finished! が表示され終了となる．

更に次回 random_check 起動時には前に書いたコードが answer.rb に格納されたままなので全て削除するのではなく，前のコードの必要な部分は残すことができる．

random_check の大きな目的は typing 速度，正確性の向上，editor 操作や Ruby 言語の習熟に重点を置いている．いかに早く終わらせるかのポイントが typing 速度，正確性と editor 操作である．

2.16 sequential_checkの使用方法

2.17 実装コードの解説

2.18 起動時のプログラム

editor_learner を起動したときに自動に動く部分である．コードは以下の通りである．

```
def initialize(*args)
      super
      @prac_dir="#{ENV['HOME']}/editor_learner/workshop"
      @lib_location = Open3.capture3("gem environment gemdir")
      @versions = Open3.capture3("gem list editor_learner")
      p @latest_version = @versions[0].chomp.gsub(' (', '-').gsub(')','')
      @inject = File.join(@lib_location[0].chomp, "/gems/#{@latest_version}/lib")
      if File.exist?(@prac_dir) != true then
        FileUtils.mkdir_p(@prac_dir)
        FileUtils.touch("#{@prac_dir}/question.rb")
        FileUtils.touch("#{@prac_dir}/answer.rb")
        FileUtils.touch("#{@prac_dir}/random_h.rb")
        if File.exist?("#{@inject}/random_h.rb") == true then
          FileUtils.cp("#{@inject}/random_h.rb", "#{@prac_dir}/random_h.rb")
        elsif
          FileUtils.cp("#{ENV['HOME']}/editor_learner/lib/random_h.rb", "#{@prac_dir}/
        end
      end
      range = 1..6
      range_ruby = 1..3
      range.each do|num|
        if File.exist?("#{@prac_dir}/ruby_#{num}") != true then
          FileUtils.mkdir("#{@prac_dir}/ruby_#{num}")
          FileUtils.touch("#{@prac_dir}/ruby_#{num}/q.rb")
          FileUtils.touch("#{@prac_dir}/ruby_#{num}/sequential_h.rb")
          if File.exist?("#{@inject}/sequential_h.rb") == true then
            FileUtils.cp("#{@inject}/sequential_h.rb", "#{@prac_dir}/ruby_#{num}/seque
          else
            FileUtils.cp("#{ENV['HOME']}/editor_learner/lib/sequential_h.rb", "#{@prac
```

```
        end
        range_ruby.each do|n|
          FileUtils.touch("#{@prac_dir}/ruby_#{num}/#{n}.rb")
        end
      end
    end
  end
```

この部分は基本的にディレクトリやファイルの作成が主である．上から順に説明すると，@prac_dir はホームディレクトリ/editor_learner/workshop を指しており，ファイルを作る際のパスとして作成されたインスタンス定数である．その後の 3 つのインスタンス定数 (@lib_location,@versions,@latest_version) は gem で install された場合ファイルの場所がホームディレクトリ/.rbenv/versions/2.4.0/lib/ruby/gems/2.4.0/gems の editor_learner に格納されているため gem で install した人と github で install した人とではパスが変わってしまうためこれらの 3 つのインスタンス定数を用意した．実際の振る舞いとしては，File.exist により prac_dir がなければディレクトリを作成しさらにその中に question.rb と answer.rb を作成する．gem にリリースしていることから gem で install した人と github で install した人のパスの違いを if 文で条件分岐させている．これにより random_h.rb を正常にコピーすることができた．

2.19 delete

```
desc 'delete [number~number]', 'delete the ruby_file choose number to delet\
e file'

def delete(n, m)
  range = n..m
  range.each{|num|
  if File.exist?("#{@prac_dir}/ruby_#{num}") == true then
    system "rm -rf #{@prac_dir}/ruby_#{num}"
```

```
  end
  }
end
```

2.20 random_check

```
desc 'random_check', 'ramdom check your typing and edit skill.'
    def random_check(*argv)
      random = rand(1..15)
      p random
      s = "#{random}.rb"
      puts "check starting ..."
      puts "type following commands on the terminal"
      puts "> emacs question.rb answer.rb"

      src_dir = File.expand_path('../..', __FILE__) # "Users/souki/editor_learner"
      if File.exist?("#{@inject}/random_check_question/#{s}") == true then
        FileUtils.cp("#{@inject}/random_check_question/#{s}", "#{@prac_dir}/question.r
      elsif
        FileUtils.cp(File.join(src_dir, "lib/random_check_question/#{s}"),  "#{@prac_d
      end
      open_terminal

      start_time = Time.now
      loop do
        a = STDIN.gets.chomp
        if a == "check" && FileUtils.compare_file("#{@prac_dir}/question.rb", "#{@prac
          puts "It have been finished!"
          break
```

```
      elsif FileUtils.compare_file("#{@prac_dir}/question.rb", "#{@prac_dir}/answer
        @inputdata = File.open("#{@prac_dir}/answer.rb").readlines
        @checkdata = File.open("#{@prac_dir}/question.rb").readlines
        diffs = Diff::LCS.diff("#{@inputdata}", "#{@checkdata}")
        diffs.each do |diff|
          p diff
        end
      end
    end
    end_time = Time.now
    time = end_time - start_time - 1

    puts "#{time} sec"
  end
```

2.21 sequential_check

```
desc 'sequential_check [lesson_number] [1~3number] ','sequential check your typing ski
  def sequential_check(*argv, n, m)
    l = m.to_i - 1

    @seq_dir = "lib/sequential_check_question"
    q_rb = "ruby_#{n}/#{m}.rb"
    @seqnm_dir = File.join(@seq_dir,q_rb)
    @pracnm_dir = "#{ENV['HOME']}/editor_learner/workshop/ruby_#{n}/#{m}.rb"
    @seqnq_dir = "lib/sequential_check_question/ruby_#{n}/q.rb"
    @pracnq_dir = "#{ENV['HOME']}/editor_learner/workshop/ruby_#{n}/q.rb"
    @seqnl_dir = "lib/sequential_check_question/ruby_#{n}/#{l}.rb"
    @pracnl_dir = "#{ENV['HOME']}/editor_learner/workshop/ruby_#{n}/#{l}.rb"
```

```
puts "check starting ..."
puts "type following commands on the terminal"
src_dir = File.expand_path('../..', __FILE__)
if File.exist?("#{@inject}/sequential_check_question/ruby_#{n}/#{m}.rb") == true
  FileUtils.cp("#{@inject}/sequential_check_question/ruby_#{n}/#{m}.rb", "#{@pra
elsif
  FileUtils.cp(File.join(src_dir, "#{@seqnm_dir}"),  "#{@pracnq_dir}")
end
if l != 0 && FileUtils.compare_file("#{@pracnm_dir}", "#{@pracnq_dir}") != true
  FileUtils.compare_file("#{@pracnl_dir}", (File.join(src_dir, "#{@seqnl_dir}"))
  FileUtils.cp("#{@pracnl_dir}", "#{@pracnm_dir}")
end

if FileUtils.compare_file(@pracnm_dir, @pracnq_dir) != true then
  system "osascript -e 'tell application \"Terminal\" to do script \"cd #{@prac
  loop do
    a = STDIN.gets.chomp
    if a == "check" && FileUtils.compare_file("#{@pracnm_dir}", "#{@pracnq_dir}"
      puts "ruby_#{n}/#{m}.rb is done!"
      break
    elsif FileUtils.compare_file("#{@pracnm_dir}", "#{@pracnq_dir}") != true the
      @inputdata = File.open("#{@pracnm_dir}").readlines
      @checkdata = File.open("#{@pracnq_dir}").readlines
      diffs = Diff::LCS.diff("#{@inputdata}", "#{@checkdata}")
      diffs.each do |diff|
        p diff
      end
    end
  end
```

```
      else
       p "ruby_#{n}/#{m}.rb is finished!"
      end
    end
```

2.22 open_terminal

新しいターミナルを開くメソッドである．コードは以下の通りである．

```
def open_terminal
        pwd = Dir.pwd
        system "osascript -e 'tell application \"Terminal\" to do script \"cd #{@prac
      end
```

新しく開かれたターミナルは prac_dir(editor_learner/workshop) のディレクトリからスタートするように設定されている．

2.23 他のソフトとの比較

2.24 総括

ここには他のタイピングソフトとのひか k