2018年2月16日
八重樫祐喜

技術説明資料

0.　目次

1.　概要

1. 本プログラムはコンソール上で動作する住所レコード検索アプリケーションである。
   また、検索アルゴリズムはn=2のN-Gramインデックス方式を採用した。

2. 始めに住所レコードが一覧されたデータソースファイルを読み込み住所データを作成する。
   この時、コンソールに`住所データファイルを読み込み中です..`と出力。

3. インデックスファイルの有無を確認し、

   3.a. 無ければ住所データよりインデックスデータを作成して、
   インデックスファイルとして所定のディレクトリに出力する。
   この時、コンソールに`インデックスファイルを作成中です..`と出力。

   3.b. あれば、インデックスファイルを読み込んで、インデックスデータを作成する。
   この時、コンソールに`インデックスファイルを読み込み中です..`と出力。

4. コンソールに`入力:`と出力し、ユーザーは検索する文字列を入力する。

5. コンソールに`出力:`と出力する。

6. ユーザーが入力した文字列の最初の文字をインデックスデータより検索し、該当するデータのデータソースにおける行番号群を取得する。

7. 行番号群を元に住所データから該当する住所レコード群を取得する。

8. 住所レコード群からユーザーが入力した文字列のすべてが含まれる住所レコード群を抽出する。

9. 抽出した住所レコード群を一行ずつコンソールに出力する。

10. 概要4-8を繰り返し、概要4でユーザーが`quit`と入力した時アプリケーションを終了する。
    終了の際にはコンソールに`終了します`と出力する。

2. データソースファイル

1. 日本全国の郵便番号と住所を対応させたCSV形式のファイル。
2. http://www.post.japanpost.jp/zipcode/dl/kogaki/zip/ken_all.zipよりダウンロードすることができる。
3. ダウンロードしたファイルを次のファイル名で保存した。

   > search_address/download/KEN_ALL.csv

4. データの項目は次の表の通り。

列数	項目	形式
1	全国地方公共団体コード	半角数字
2	（旧）郵便番号（5桁）	半角数字
3	郵便番号（7桁）	半角数字
4	都道府県名	半角カタカナ（コード順）
5	市区町村名	半角カタカナ（コード順）
6	町域名	半角カタカナ（五十音順）
7	都道府県名	漢字（コード順）
8	市区町村名	漢字（コード順）
9	町域名	漢字（五十音順）
10	一町域が二以上の郵便番号で表される場合の表示	"1"は該当、"0"は該当せず
11	小字毎に番地が起番されている町域の表示	"1"は該当、"0"は該当せず
12	丁目を有する町域の場合の表示	"1"は該当、"0"は該当せず
13	一つの郵便番号で二以上の町域を表す場合の表示	"1"は該当、"0"は該当せず
14	更新の表示	"0"は変更なし、"1"は変更あり、"2"廃止 （廃止データのみ使用）
15	変更理由	"0"は変更なし、"1"市政・区政・町政・分区・ 政令指定都市施行、　"2"住居表示の実施、"3"区画整理、"4"郵便区調整等、"5"訂正、　"6"廃止（廃止データのみ使用）

3.　インデックスファイル

1. データソースファイルからn=2のN-Gramインデックス形式に基づいて作成されたYAML形式のファイル。
2. データの構造は以下の通り。
   2.1. データソースファイルの各レコードごとに、項目7.都道府県名、8.市区町村名、9.町域名を連結し、
   そこから隣り合った2文字の組み合わせのそれぞれをキーとする。
   2.2. 2.1でキーとした文字の組み合わせに対する値として、
   その文字の組み合わせが存在するデータソースファイル中の行番号の配列を格納する。
   2.3. データソースファイルからインデックスファイルを作成する例を表3-2,リスト3-2に示す
3. プログラムが次のファイル名で自動保存する。

> search_address/output/search_index.yaml

表3-2: データソース例

行番号	データ
107	八重山
108	八重桜
109	八重樫

リスト3-2: 表3-2のデータソース例から作成したインデックス例

```
八重:
- 107
- 108
- 109
重山:
- 107
重桜:
- 108
重樫:
- 109
```

4.　検索アルゴリズムについて N-Gram

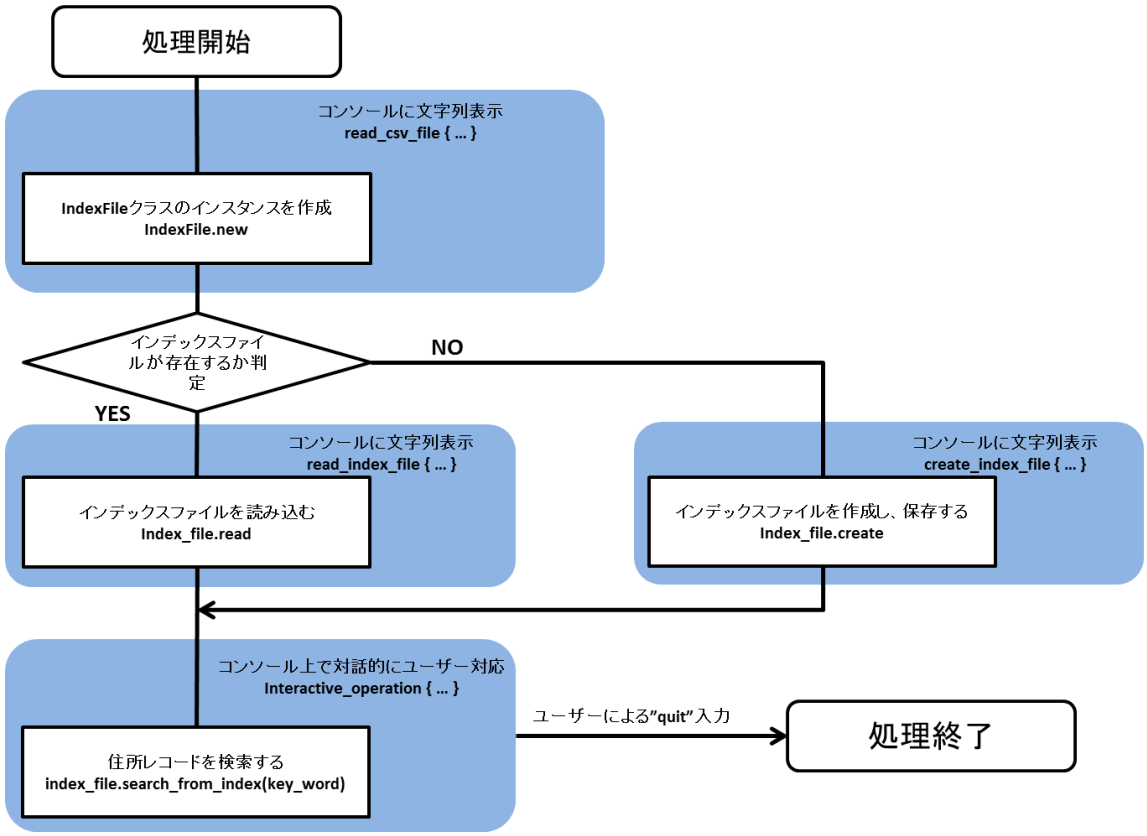
1. 本プログラムでは、n=2のN-Gramインデックス方式の検索方式を採用した。
2. データソースからインデックスを作成する方法は、「3.　インデックスファイル」の通り。
3. 作成したインデックスを使って検索結果を抽出する方法を以下に示す。
   3.1. ユーザーの入力した検索文字列のうち、最初の1文字目を取り出す。
   3.2. インデックスのキーから、3.1で取り出した文字を含むキーを検索し、対応する値（配列）を取得する。
   該当するキーが複数存在する場合は、そのそれぞれに対応する値（配列）の和集合（配列）を取得する。
   3.3. 3.2で取得した値（配列）は行番号の一覧になっているので、それらに対応するデータソースのレコードを取得する。
   3.4. 取得したデータソースのレコードの特定の項目から、
   ユーザーの入力した検索文字列のすべての文字[*1]が含まれているレコードを抽出し完了。

※1 ユーザーが入力した検索文字列の最初の文字が含まれていることは3.2より明確なので、実際は2文字目以降の文字の存在有無をチェックしている。

5. 動作

5.1. トップレベルrunメソッド内のフローチャート

以下にトップレベルのメソッドである、runメソッド内のフローチャートを示す。
なお、フローチャート内のメソッドについては、5.3で説明する。



5.2. 定数

以下に定数の一覧を示す。

定数名	内容	値
CSV_FILE_PATH	データソースファイルのパス	search_address/download/DEN_ALL.csv*2
INDEX_FILE_PATH	インデックスファイルのパス	search_address/output/search_index.yaml*2
COLUMN_POSTCODE	住所レコードの項目番号（郵便番号）*3	2
COLUMN_PREFECTURES	住所レコードの項目番号（都道府県名）*3	6
COLUMN_CITY	住所レコードの項目番号（市区町村名）*3	7
COLUMN_TOWN	住所レコードの項目番号（町域名）*3	8
COLUMN_OVER_TOWN_FLAG	住所レコードの項目番号 （一つの郵便番号で 二以上の町域を表す場合の表示）*3	12
NOT_APPLICABLE	住所レコードの特定項目で使われる値 （該当せず）	"0"

※2 実際は絶対パスを格納しているが、便宜上ここではsearch_address以降を記述している。

※3 項目番号の始まりを0としてカウントした数値をしている。

5.3. 各クラス・モジュールの機能

5.3.1. SearchAddressモジュール

プログラムのメインとなるトップレベルのモジュール

■ runメソッド

□ 引数： なし
□ 説明： トップレベルのメソッド。
詳細は5.1 トップレベルrunメソッド内のフローチャートを参照。

5.3.2. Defineモジュール

プログラム共通のグローバル定数を定義するモジュール。
定数の一覧は5.2 定数を参照。

5.3.3. Managerモジュール

ユーザーにインターフェースを提供するモジュール

■ read_csv_fileメソッド

□ 引数： (&block) 住所データファイルの読み込みを行うブロック
□ 説明： 引数で受け取ったブロックを処理している間、
コンソールに「住所データファイルを読み込み中です..」を表示し続ける。
実際には表示する文字列と引数で受け取ったブロックをManager#setup_threadに渡している。

■ read_index_fileメソッド

□ 引数： (&block) インデックスファイルの読み込みを行うブロック
□ 説明： 引数で受け取ったブロックを処理している間、
コンソールに「インデックスファイルを読み込み中です..」を表示し続ける。
実際には表示する文字列と引数で受け取ったブロックをManager#setup_threadに渡している。

■ create_index_fileメソッド

□ 引数： (&block) インデックスファイルの作成を行うブロック
□ 説明： 引数で受け取ったブロックを処理している間、
コンソールに「インデックスファイルを作成中です..」を表示し続ける。
実際には表示する文字列と引数で受け取ったブロックをManager#setup_threadに渡している。

■ interactive_operationメソッド

□ 引数： ( ) 検索・出力の処理を行うブロック
□ 説明： コンソールに「入力：」を表示し、ユーザーからの検索キーワードの入力を待つ。
受け取った検索キーワードが、
a) nilの場合は、プログラムを終了。
受け取った検索キーワードがnilじゃなければ、
空白、改行を削除してそれぞれの文字を要素として配列を作成。
作成した配列が、
b) ["q", "u", "i", "t"]の場合は、プログラムを終了、
c) 空の場合は、再度コンソールに「入力：」を表示し次の検索キーワードの入力を待つ、
d) 上記a,b,c以外の場合は、次の処理へ。
コンソールに「出力：」を表示し、作成した配列を引数で受け取ったブロックに渡して処理する。

■ get_key_wordsメソッド（private）

□ 引数： (input) ユーザーが入力した検索キーワード
□ 説明： 入力された検索キーワードから改行、空白を取り除いてUTF-8でエンコードする。
最後に一文字ずつ分割して配列にして返り値として返す。

■ **setup_threadメソッド（private）**

□ 引数： (output) コンソールに表示する文字列
( ) コンソールの表示と並行して内部的な処理をするブロック
□ 説明： スレッドを作成してそこでコンソールに引数outputを表示する。
実際の処理はManager#start_threadにoutputを渡して行っている。
スレッドを作成した後に、引数で受け取ったブロックを処理。
ブロックの処理が終わったら、作成したスレッドを終了し、
このメソッドの返り値として処理したブロックの返り値を返す。

■ **start_threadメソッド（private）**

□ 引数： (output) コンソールに表示する文字列
(sleep_time=1) コンソールの表示を再表示する間隔（秒）
□ 説明： スレッドを作成してコンソールに文字列を表示。
表示する文字列は1秒ごとに再表示をし、
表示末尾の「...」を増減させて処理中であることを明示する。
このメソッドの呼び出し元でスレッドを終了する為に、
作成したスレッドのインスタンスを返り値として返す。

■ **kill_threadメソッド（private）**

□ 引数： (thread) 終了するスレッドのインスタンス
□ 説明： 引数を元にスレッドを終了する。

■ **exit_searchメソッド（private）**

□ 引数： なし
□ 説明： コンソールに「\n終了します」を表示し、プログラムを終了する。

5.3.4. IndexFileクラス

インデックスファイルを操作するクラス。

□ インスタンス変数： (@data) インデックスデータを格納する。
(@separated) 住所レコードの連結が必要なデータを格納する。

■ exist?メソッド（クラスメソッド）

□ 引数： なし
□ 説明： インデックスファイルの存在を確認。
true/falseを返す。

■ initializeメソッド

□ 引数： なし
□ 説明： IndexFileクラスのインスタンスを生成。
インスタンス変数@dataを空の配列で値を初期化するハッシュで初期化。
インスタンス変数@separatedを空の配列で値を初期化するハッシュで初期化。
Searchモジュールのモジュール変数@@csvがnilの場合、
Search#read_csv_fileを呼び出して@@csvにデータソースファイルのデータを格納する。

■ createメソッド

□ 引数： なし
□ 説明： Searchモジュールのモジュール変数@@csvから、
インデックスデータ、要連結住所レコードデータを作成しそれぞれ@data、@separatedに格納。
この際、インデックスデータはn=2のN-Gramインデックス形式を
String#to_ngram_indexにて作成している。
作成したインデックスデータからインデックスファイルを作成する。
要連結住所レコードデータの作成はIndexFile#create_separated_dataで、
インデックスデータはIndexFile#create_separated_dataに渡したブロック内で作成する。
インデックスファイルの作成はIndexFile#create_index_fileで行う。

■ readメソッド

□ 引数： なし
□ 説明： インデックスファイルから読み込んだインデックスデータを@dataに格納。
要連結住所データを作成し@separatedに格納。

■ create_index_fileメソッド（private）

□ 引数： なし
□ 説明： @dataからインデックスファイルを作成する。

■ read_index_fileメソッド（private）

□ 引数： なし
□ 説明： インデックスファイルからインデックスデータを作成し、返り値として返す。

■ read_csv_fileメソッド（private）

□ 引数： なし
□ 説明： データソースファイルが存在する場合、
データソースファイルからUTF-8でエンコーディングしたCSV形式のデータを読み込み、
Searchモジュールのモジュール変数@@csvに格納する。
格納の形式はCSVデータの各行の各列の値を値とする配列の配列。
データソースファイルが存在しない場合は、
SearchAddressErrorを発生させ、メッセージと共にプログラムを終了する。

■ **create_separated_dataメソッド（private）**

□ 引数： ( ) 住所レコードとその行番号を受け取るブロック

□ 説明： 住所レコードデータから要連結住所レコードを作成して@separatedに格納する。
Searchモジュールのモジュール変数@@csvに格納されている住所レコードデータから、
一行ずつ住所レコードとその行番号を取得して処理。
住所レコードから郵便番号、住所を取得し、要連結住所レコードを作成している。
このメソッドに引数としてブロックが与えられている場合は、
ブロックに住所レコードと行番号を渡して処理。

5.3.5. Searchモジュール

ユーザーに入力した文字列から住所レコードを検索するモジュール。

□ モジュール変数： (@@csv) 住所レコードデータを格納する。

■ search_from_indexメソッド

□ 引数： (key_words) 空白・改行を取り除いたユーザー入力検索文字列の一文字ずつを値とした配列

□ 説明： ユーザーが入力した検索文字列から住所レコードを検索する。
検索文字列の最初の文字を、
IndexFileクラスのインスタンス変数@dataに格納のインデックスデータから検索し、
当該最初の文字を含む住所レコードの行番号を取得する。
取得した行番号から該当の住所レコードを抽出し、
さらに検索文字列のその他の文字のすべてを含む住所レコードを絞り込む。
その結果取得できた住所レコードの郵便番号と住所の列から文字列を作成し、
1行ずつコンソールに出力する。
検索文字列の最初の文字以外のすべての文字を含む住所レコードの絞り込みとその結果の出力は、
Search#search_from_csvにて行っている。

■ search_from_csvメソッド（private）

□ 引数： (row_numbers) インデックスデータの検索から取得した住所レコードデータの行番号の配列
(key_words) ユーザー入力検索文字列から最初の文字を取り除き、それぞれの文字を値とした配列

□ 説明： 住所レコードデータを受け取った引数row_numbersで検索して、
その結果の住所レコードに受け取った引数のkey_wordsのすべてを含むかをチェックする。
すべて含まれている住所レコードは、その郵便番号、住所をコンソールに出力する。

■ get_postcode_and_addressメソッド（private）

□ 引数： (row) 住所レコードデータ

□ 説明： 住所レコードデータから郵便番号、住所を取得し、返り値として返す。

5.3.6. その他

■ String#to_ngram_index

□ 引数： (n) N-Gramインデックス形式で作成するインデックスの文字数
( ) インデックスとして作成した各文字列を受け取るブロック

□ 説明： レーシーバの文字列から隣り合ったn文字の組み合わせを作成して、ブロックに渡す。

6. テスト

1. テストはrspecを使用して行う。
2. テストスクリプトは、`search_address/lib/`以下の各ファイルごとに作成し、次のディレクトリ以下に置く。

   > search_address/spec/