

Tabelog System 設計書

1. 全体の概要

- システム名: Tabelog System
- 目的: Tabelog(食べログ)データを収集・整形・分析し、現在位置からの距離や価格帯を考慮して最適なお店を可視化する
- 主な処理フロー:

*****処理の前にdataフォルダは存在すれば消す*****

1. データ収集(tabelog_data_collect.py)
2. データ整形(tabelog_data_shaping.py)
3. 現在位置計算(calculate_location.py / tabelog_data_shaping.py)
4. 最終点数を計算(analysts.py)
5. 地図生成(generate_map.py)

2. データ収集: tabelog_data_collect.py

2.1 目的

食べログのWebページから店舗情報を取得

対象データ: 店名、住所、評価、価格帯、カテゴリなど

2.2 入力

```
areas = ["北千住", "南流山"]
```

```
menus = ["スペイン料理", "刀削麺"]
```

検索対象エリアと料理ジャンルを指定

エリア・メニューの組み合わせごとに処理

→例: 北千住 × スペイン料理

2.3 出力

データを辞書形式で格納

```
[  
    {'星5段階評価': '3.34', '店名': 'Cabana (カバナ)', ... },  
    {'星5段階評価': '3.28', '店名': 'スペインバル ソルマヨール', ... },  
    ...  
]
```

最終的にdata/source_data\検索駅名\食べ物.csvでcsvファイルとして保存。

<CSVに出力する理由>

永続化できる → スクレイピングで取得したデータを後で再利用できる

中間データとして他処理でも使いやすい → 例えば、別の分析や地図生成に流用できる

デバッグしやすい → CSVを開けばスクレイピング結果の確認が簡単

処理が分離できる → データ収集と加工処理を別ステップにできる

メモリ使用量が増えるのを抑える → 大量データだとRAMを圧迫してしまう

カラム名	説明
店名	店舗名
ジャンル	食べログのカテゴリ（複数可）
星5段階評価	食べログ評価
予約・お問い合わせ	予約用電話番号など
予約可否	「予約可」「予約不可」
住所	店舗住所
交通手段	最寄駅・徒歩距離など
営業時間	営業時間・定休日
Dinner	ディナーの価格帯
Lunch	ランチの価格帯
支払い方法	カード、電子マネー、QR決済可否
領収書（適格簡易請求書）	インボイス対応の有無
サービス料・チャージ	席料・サービス料など

2.4 主な処理

1. スクレイピング対象のお店のURLリストの作成
2. ページの取得(requests / Selenium)
3. HTML解析・情報抽出
4. CSV/JSONへの書き出し

3. データ整形:tabelog_data_shaping.py

3.1 目的

収集した生データを分析・計算・可視化用に整形

3.2 入力

①tabelog_data_collect.py で生成されたCSV

②現在地

```
current_location = {  
    "name": "現在地",  
    "latitude": 35.834774,  
    "longitude": 139.912964,  
}
```

③何人がどの料理に投票したか

```
voice_force = {"スペイン料理": [3], "刀削麺": [5, 2]}
```

→例: 佐藤さん(の決定権のレベル3)がスペイン料理に投票した

加藤さん(の決定権のレベル5)が刀削麺に投票した

斎藤さん(の決定権のレベル2)が刀削麺に投票した

④決定権のレベルを無視して単純な多数決で決めるかと

より強く決定権のレベルの影響を強めるか

```
alpha_value = 0.5
```

(0に近いほど多数決、1に近いほど決定権のレベルを重視)

3.3 出力

整形済みCSV/DataFrame(必要なカラムのみ保持)

カラム名	内容・説明
星5段階評価	店舗の評価(小数、例:3.34)
店名	店舗の名前(例:Cabana(カバナ))
住所	店舗の住所(例:東京都足立区千住3-68-20)
Dinner	ディナーの平均価格(整数、円)
Lunch	ランチの平均価格(整数、円)
項目	CSV ファイル元のカテゴリーやジャンル名 (例:スペイン料理、刀削麺)
緯度	店舗の緯度(例:35.751335)
経度	店舗の経度(例:139.803787)
現在地からの 距離(km)	指定した現在地から店舗までの距離(km、浮動小数点)

現在地点からの徒歩(分)	上記距離を徒歩で移動した場合のおおよその時間(分、浮動小数点)
声の大きさ	投票結果に基づく項目ごとのスコア(数値、alpha パラメータを考慮)

3.4 主な処理

不要カラムの削除

データ型変換(文字列 → 数値、日付など)

価格帯の平均値の計算など項目ごとの調整

欠損値処理

各ファイルの連結

4. 現在位置計算: `calculate_location.py` / `tabelog_data_calculate.py`

4.1 目的

- 店舗の住所から緯度・経度を(API)
msearch.gsi.go.jp/address-search/AddressSearch?q=使って算出
- 現在地・お店の緯度経度から距離・徒歩時間を計算
- `tabelog_data_calculate`に計算したデータを渡す

4.2 入力

- 移動距離の項目を付け足したいcsvファイル

今回は3, で作成したcombined_data.csvファイル

- 現在位置の緯度・経度

```
current_location = {  
    "name": "現在地",  
    "latitude": 35.834774,  
    "longitude": 139.912964,  
}
```

4.3 出力

- 各店舗までの距離・徒歩時間を含むDataFrame

4.4 主な処理

1. 緯度・経度変換 (Geocoding)
2. 距離計算 (Haversine式など)
3. 徒歩時間計算 (距離 / 平均歩行速度)

5. 最終点数を計算: analysts.py

5.1 目的

- 現在位置と各店舗の情報(評価・価格・徒歩時間等)を元に
- 各店舗の最終点数を算出し、ランキング可能なデータを出力

5.2 入力

- 前処理済みCSV(例:`data/maked_data/combined_data.csv`)

- デイナーかランチか指定(例 "`lunch`", "`dinner`")

- 許容価格上限、許容徒歩距離(分)

- 重み付け(例)

1. `weight = {"distance":1.1,"budget":1.2,"evaluate":1.3}`

5.3 出力

- 点数計算済みCSV(例:`data/result_data/result_data.csv`)
- 出力カラム例: 点数、店名、星5段階評価、価格、項目、徒歩(分)、緯度、経度

5.4 処理フロー

1. データ読み込み
2. フィルタリング(最大価格・最大徒歩時間)
3. 逆数変換(価格・徒歩時間を1/xに変換)
4. 正規化(0.1~1)
5. 重み付け適用
6. 点数合算・降順ソート
7. CSV出力
- 2.

6. 地図生成:`generate_map.py`

6.1 目的

- 現在位置と各店舗を地図上に可視化し、ピンを立てる

6.2 入力

- 点数計算済みDataFrame(例:`data/result_data/result_data.csv`)
- 現在地情報

例:

```
current_location = {"name": "現在地",  
                    "latitude": 35.64338, "longitude": 139.66952}
```

6.3 出力

- ピン付き地図(HTML / Folium / Google Maps API)

6.4 主な処理

1. 地図初期化
2. 現在地マーカ―設置
3. 各店舗マーカ―設置
4. 地図保存 / 表示