1Y-05

インメモリデータベースを用いた高速逆ジオコーダ

荒巻　凌[†§]　　星野崇宏[‡§]

慶應義塾大学 商学部[†]　　慶應義塾大学 経済学部[‡]　理化学研究所 革新知能統合研究センター(AIP)[§]

1. 背景

位置情報を用いたマーケティングの隆盛に伴い、データ分析の現場においては位置情報ビッグデータを解析する機会が増加している。解析対象データは単にサイズが大きいだけでなく、利用者のプライバシーに関わる情報であり、慎重な取り扱いを要するものである。

本研究では、膨大な位置情報データを高速かつセキュアに処理するためのコマンドラインツール“tadataka”を開発した。

https://github.com/hoshinolab/tadataka

2. 既存手法の問題

座標（緯度・経度）を住所に変換する逆ジオコーディング（Reverse Geocoding）を実施するにあたり、一般的に用いられる手法はオンライン地図サービス等の外部 API に HTTP リクエストとして座標データを送信するものである。これは手軽ではあるものの、主に 2 つの問題がある。

2.1 件数の上限

API サービスは有償・無償を問わず、処理可能な件数に上限が設けられている。小規模であれば問題ないが、ビッグデータ処理の現場においてはこれが大きな障壁となる。

実際、筆者の所属する研究室においては、企業との共同研究に際して 15TB（数百億件のレコード）を超えるデータを取り扱っており、これらを逆ジオコーディングする際に API を利用することはそもそも困難であるといえる。

2.2 セキュリティ・コンプライアンス上の問題

位置情報データはセンシティブなものであり、利用者から「外部に送信しない」事を条件として得ることが多い。外部 API を用いてこのデータを解析することは「外部への送信」に該当する可能性が高く、利用者に対する契約違反になるといえる。

また、外部 API を提供する企業が悪意を持ってデータを流用するリスクや、ツールや通信経路の脆弱性により情報が流出する可能性も否定できない。

本研究では 2 つの問題を解決するため、オンプレミス、つまり解析環境を手元に置く位置情報データ解析ツールを構築する。

3. 使用データおよび実装手法

逆ジオコーディングには座標データを照らし合わせる住所データが必要となる。本研究においては国土交通省国土政策局が公開する街区レベル位置参照情報（以下、位置参照情報）および国土地理院が公開する電子国土基本図（地名情報）「住居表示住所」（以下、住居表示住所）を利用した。どちらとも住所と座標のデータが格納されており、与えられた座標と最も近いデータを逆ジオコーディングの結果として扱う。

住居表示住所はきめ細かく「号」レベルまでのデータが整備されているが、一部の市区町村のみで提供されている。一方で位置参照情報は全国をカバーしているが、番地ごとの代表点しかデータがない。そこで、住居表示住所を優先して探索し、該当データがない場合に位置参照情報のデータを利用する形とした。

探索時、全国のデータに総当たりしていると計算量が膨大になる。それゆえ、探索するにあたり、データを細かく分割する必要がある。地理空間上をグリッドに分割する手法は複数あるが、今回は Google が提唱している Open Location Code(OLC)を採用した。OLC を用いると、座標を一意の文字列に変換することができる。

High speed reverse geocoder using an in-memory database
† Keio University Faculty of Business and Commerce
‡ Keio University Faculty of Economics
§ RIKEN Center for Advanced Intelligence Project (AIP)

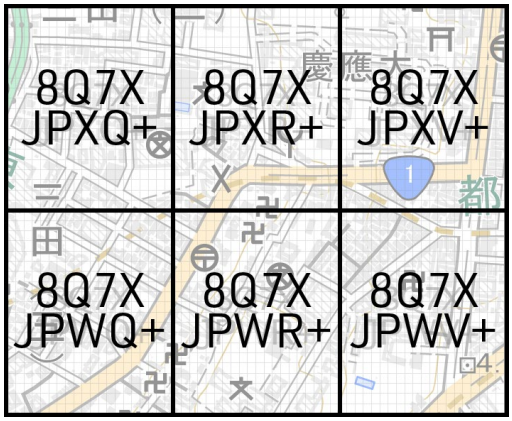

図 1:港区三田付近の OLC グリッド
（背景地図は地理院タイルを利用）

これを用い、8Q7XJPXR+FX と変換される位置情報に対しては 8Q7XJPXR+のグリッドに含まれる住所から最近傍のものを見つけ、それを住所とみなす。

また、使用するデータはインメモリデータベースである Redis に格納する。Redis には OLC のグリッド（8Q7XJPXR+等）をキーとして list 型で格納している。Redis はデータベースエンジンの中でも高速であることが知られており、処理内容にもよるがマイクロ秒単位で保存されているデータの取得を行うことができる。

4. パフォーマンス測定

ベンチマークとして、「駅データ.jp」で公開されている日本全国の駅の座標リストから番地単位の住所への変換を行った。2019 年 9 月 28 日のデータで 10847 件の駅が収録されている。ならびに、企業から提供を受けた 36990279 件のスマートフォン由来の位置情報データを分析した。

Ryzen 9 3900X + 64GB RAM の Windows 10 環境上で Windows Subsystem for Linux を利用し、Go 1.13 で tadataka をビルドした。また、Redis ver 4.0.9 を使用した。

比較対象として、Python で一般的なジオコーディングライブラリ geocoder から OpenStreetMap の API を呼び出す処理で同等の処理を行った。なお、OSM は正確に取得できない住所に対しては都道府県や市区町村レベルの住所を返す。

tadataka を用いたジオコーディングの処理速度は以下の通りである。（time コマンドによる計測）

件数	所要時間(real)	取得失敗
10847	0m2.907s	1894 （約 17.4%）
36990279	115m11.699s	4706441 件 （約 12.7%）

従来手法（Python で geocoder(OSM)使用）は以下の通りである

件数	所要時間(real)
10847	223m41.916s

精度に関しての情報や、他の実行環境などにおける処理速度の比較など、詳細なパフォーマンスは発表時に公表する

5. 今後の課題

データにもよるが、現時点では 1 割～2 割程度の座標で住所を取得することができない。これらは住居表示住所と位置参照情報がカバーしていないエリア（グリッド）のデータであるが、隣接するグリッドに最近傍の住所が存在している可能性がある。それゆえ、隣接するグリッドの探索も行うことにより、より高精度の逆ジオコーディングが可能になると考えられる。

また、今回は逆ジオコーディングを実装したが、位置情報データの処理においてはジオコーディング（住所から座標への変換）の需要も存在する。座標と異なり、入力値が文字列であることから表記ゆれが存在し、たとえば「東京都港区三田 2 丁目 15−45」と「東京都港区三田二丁目十五番四十五号を同一のものとして扱うといった、住所の正規化が必要となる。正規化が可能であれば、今回のインメモリデータベースを用いる手法を用いてジオコーダの実装にも取り組みたいと考えている。

6. 参考文献

Doug Rinckes, Philipp Bunge: Open Location Code: An Open Source Standard for Addresses, Independent of Building Numbers And Street Names (GitHub) from https://github.com/google/open-location-code/blob/master/docs/olc_definition.adoc (accessed 2020-01-09)