Automation of intermediate geocoding for web GIS applications using cloud functions

Bahniuk M.A, Krasovska I.G.
(National Aerospace University named after M.E. Zhukovsky «KHAI», E - mail: martin.bahniuk@gmail.com, ines75ma@ukr.net)

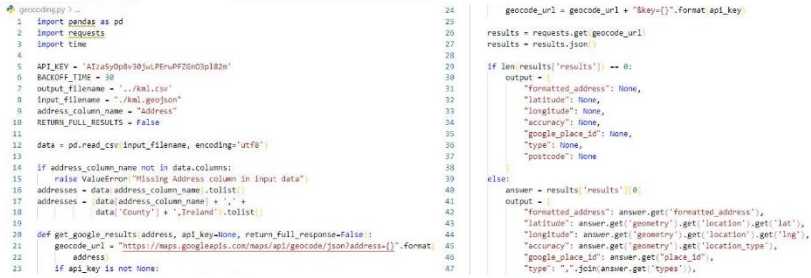
Location data plays one of the main roles in the integration of information about society and the environment. The use of up-to-date cartographic information presented in an interactive and easy-to-understand format influences personal perspective and provides more context for decision-making regarding interaction with government institutions and public enterprises and organizations.

To solve this goal, there is a need for analysis and study of relevant web portal development tools for the creation of a modern multi-functional web application and the capabilities of web GIS tools. Preference is given to scalable services, such as cloud technologies, serverless functions, as well as application programming interfaces and accompanying cloud technologies.

At the moment, there are three main providers of cloud services - AWS from Amazon, Azure from Microsoft, and Google Cloud from Google (Alphabet), respectively. These companies occupy the lion's share of the market around the world (except for China, where Alibaba Cloud is the domineering cloud provider). They are technological leaders and set trends in the development of cloud Infrastructure as a Service (IaaS) services.

For the purposes of geocoding, the process of taking a text-based description of a location, such as an address or the name of a place and returning geographic coordinates, usage of vast address libraries which every one of these providers employ is necessary. It is important to note that automation is the goal, so instead of manually consuming the returned data, a geocoding

script was written in the Python 3 programming language. As an example, NAU KHAI student's addresses were geocoded using cloud functions that ran the script automatically (fig. 1).



```
geocoding.py > ...
import pandas as pd
import requests
import time

API_KEY = 'AIzaSyDp8v30jwLPErwPFZGnO3p182m'
BACKOFF_TIME = 30
output_filename = '../kml.csv'
input_filename = "./kml.geojson"
address_column_name = "Address"
RETURN_FULL_RESULTS = False

data = pd.read_csv(input_filename, encoding='utf8')

if address_column_name not in data.columns:
    raise ValueError("Missing Address column in input data")
addresses = data.address_column_name.tolist()
addresses = data[address_column_name] + ',' +
            data['County'] + ',Ireland'].tolist()

def get_google_results(address, api_key=None, return_full_response=False):
    geocode_url = "https://maps.googleapis.com/maps/api/geocode/json?address={}".format(
        address)
    if api_key is not None:
        geocode_url = geocode_url + "&key={}".format(api_key)

    results = requests.get(geocode_url)
    results = results.json()

    if len(results['results']) == 0:
        output = {
            "formatted_address": None,
            "latitude": None,
            "longitude": None,
            "accuracy": None,
            "google_place_id": None,
            "type": None,
            "postcode": None
        }
    else:
        answer = results['results'][0]
        output = {
            "formatted_address": answer.get('formatted_address'),
            "latitude": answer.get('geometry').get('location').get('lat'),
            "longitude": answer.get('geometry').get('location').get('lng'),
            "accuracy": answer.get('geometry').get('location_type'),
            "google_place_id": answer.get("place_id"),
            "type": ",".join(answer.get('types')),
```

Figure 1. Basic Python 3 geocoding script that runs in GCloud

The result is a .csv file that has the latitude and longitude of each address, which allows for quick generation of GIS models, such as the student distribution heatmap (fig. 2).

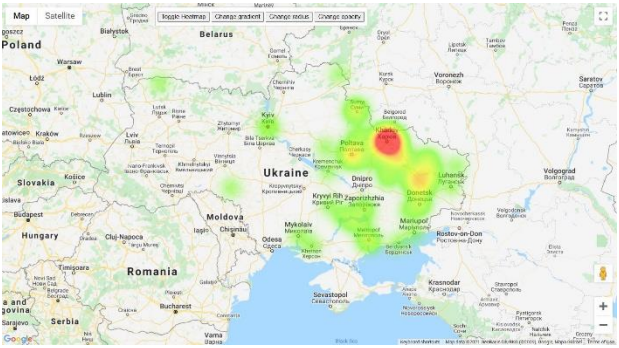


Figure 2. NAU KHAI Student distribution heatmap built with the use of automated geocoding using cloud functions

Google Geocoding API was used, which was configured in advance to isolate the address, then send it for geocoding to the application software interface. This simple method greatly speeds up GIS web application building process.