Location Identification API- User Manual

Contents

Introduction

The goal of the API is to identify the location of the project headquarters. The location identification algorithm assumes that the main location of the project is the one that has several pieces of information related to it. For example, multiple cities, airports, train stations, etc.

The algorithm uses the Nominatim[1] API to obtain information about a location and it depends on OpenStreetMaps (OSM) API.

Algorithm Workflow - Logic

The algorithm has the following steps to find the location of a project:

1- Load the all the web pages of a given project from the MongoDB based on the Project ID and the collection name
2- Reduce the number of the project webpages to the home page and first-level pages (e.g. home/about.html, home/project.html, etc.) OR the shortest 10 URLs on the project. The latter case is considered when a project website does not have first-level pages.
3- A Named Entity Recognition (NER) is used to identify Location names, i.e. "LOC" tag. We used the State-of-the-art method for location identification using BERT[2].
4- For each location, we query Nominatim API to get information about the location, like: if the name is a valid location name, the type of the location, location importance, and the candidate counties or cities of that location, the longitude and latitude of the location, the three letters ISO code of the location country.
5- Using the NER, the identified locations are split into "countries" and "cities" based on the location information, as each location has a "type".
6- Then, each city is correlated with the candidate country(ies). For example, Glasgow could be correlated with the UK and the US, each with an importance score assigned by the API.
7- From the collected location, the algorithm builds a weighted directed graph (G), of nodes and edges. The nodes refer to the countries or cities. A directed edge is created from a city node to a country node if they are correlated (a city exists in a country).
8- Each node of type "country" is weighted based on:
    a. Its appearance frequency on the project text
    b. The weight of its cities
    c. The frequency of its cities
9- The county nodes are sorted based on their weights, the highest node is the most probable place. Hence, each country will receive a score.
10- Since some projects could take place under different countries, we cluster the scores of the countries, whereas countries with close scores would be a group. The algorithm considers that the project location is the cluster of countries with the highest score. For example, if a project has the following distribution: {Spain: 90, France:92, India:54, Pakistan:44, USA: 12}, the clustering algorithm would group Spain and France under one cluster. And, since their scores are

[1] https://nominatim.org/release-docs/latest/api/Overview/

[2] https://huggingface.co/flair/ner-english-ontonotes-large

the highest, they are considered the most probable countries, being Spain is the main location and France is the secondary.

11- The city is set to the highest city weight of the best countries.

API Installation

It is recommended to use a standalone Anaconda environment to avoid any conflict with other modules. However, this module has no compatibility problems with the Social Innovation Classification tool. Hence, they can be deployed using the same Anaconda environment, called ESID. The latter approach is recommended to save disk space.

Steps to install the API:

Activate Anaconda environment: *conda activate ESID*

- Navigate to the project path: *cd /data/wesam_data/LocationDetection/*
- Install the requirements (to be done only ONCE when the tool is installed for the first time): *pip install –r requirements.txt*

API Usage

To run the Location algorithm, navigate to the algorithm path, e.g. "*/data/wesam_data/LocationDetection/*".

Then you run the main.py script

```
python main.py
```

The script operates in two modes:

1- Automatic mode: the script will predict the location of MongoDB projects and insert them to MySQL database.
2- Manual mode: the script receive a list of project IDs, predict their locations and insert them to MySQL database.

This script has two optional parameters:

- **--collection**: This parameter is "optional" and holds the name of the default MongoDB collection.
- --**projects**: This parameter is "optional" and holds a path to a text file with project IDs, one ID per line.

API Integration

The API can be integrated with other tools like projects scrapper, classifier, etc.

It has three steps:

1- Get the project text from MongoDB

```
project_data = p_collector.get_text_mongodb(project_id=project_id,
collection_name=collection_name)
```

The project ID and the collection name in MongoDB.

2- Call the location API

```
loc_info = loc_detect.location_detection_logic(project_data=project_data)
```

The returned location information "*loc_info*" will have the following format for each input project:

```
loc_info = [{
    'city_name': 'Fintry',
    'city_type': 'village',
    'country_name': 'United Kingdom',
    'country_ISO3166-1:alpha3': 'GBR',
    'lat_city': '56.0531944',
    'lon_city': '-4.223376',
    'lat_country': '54.7023545',
    'lon_country': '-3.2765753',
    'wikidata_city': 'Q1026796',
    'wikidata_country': 'Q145'
}]
```

3- Insert the project to MySQL: the location information is inserted into the "ProjectLocationNew" table.

```
p_collector.insert_predictions(project_id=project_id,
location_information=loc_info)
```

Output Storage

When calling the insert prediction function, the predicted output is saved into the database.

The location information is stored in a table called "*ProjectLocationNew*", and this table has the following columns:

LocationID: auto-increment integer refers to the record number

LocationType: if a project has one location, this parameter is set to "main location". If a project has more than one location, the most probable location will be the "main location" and the other location will be the "secondary location"

Address: the new location API does not have this field, but we left it for the sake of backward compatibility

CityName: name of the city, if found

CityType: based on the location API, the type can be a city, a village, an administrative, etc.

CityWiki: wiki key of the city

CityLongitude: city longitude

CityLatitude: city latitude

CountryName: country name

CountryLongitude: country longitude

CountryLatitude: country latitude

CountryISOName: thee letters of the country name based on the ISO standard

CountryWiki: wiki key of the country

Postcode: the new location API does not have this field, but we left it for the sake of backward compatibility

PhoneContact: the new location API does not have this field, but we left it for the sake of backward compatibility

EmailContact: the new location API does not have this field, but we left it for the sake of backward compatibility

Original_idProjects: the new location API does not have this field, but we left it for the sake of backward compatibility

Projects_idProjects: the project ID

Version: for the locations recognized using Wesam's API or the new crawler developed by Roseline it is set set to V2. If the project was detected by the old algorithm (Nikola's work), it is set to V1 or V0

DataTrace: if the location was detected by Wesam's algorithm, this column is set to "*Location Predicted by WesamAlgorithm*"

Performance Analysis

We tested the location algorithm on a dataset of 4,541 projects distributed as 1,228 and 3,283 from the new and old MongoDB collections, respectively. The algorithm was able to identify correctly 904 projects from the new collection and 2073 from the old collection, which is 0.71% and 0.63%, respectively.

Nevertheless, since the algorithm depends on mentioning the name of the location on the project text, we report the results on the projects that have the location name mentioned on the text.

For the new collection, 438 projects have the location name embedding into the text, and the algorithm identified 390, which is 0.89%. On the other hand, for the old collection, 1535 projects have the location name mentioned and the algorithm identified 1395, which is 0.88%.

In total, we can say that this tool has an accuracy of 0.88% if the location is mentioned in the text and 0.65% otherwise.

	# projects	# projects with a ground truth (country or city) and has text in MongoDB	#projects with a ground truth (country Only) and mentioned in the project text	# projects with a ground truth (city Only) and mentioned in the project text	# projects have city or country in the ground truth AND the city OR the country is mentioned in project text
New Collection	2045	1258	420	52	438
Correctly detected	NA	904	380	18	390
Accuracy	NA	0.718600954	0.904761905	0.346153846	0.890410959
Old Collection	3283	3283	1138	1459	1535
Correctly detected	NA	2073	1022	940	1359
Accuracy	NA	0.631434663	0.898066784	0.644276902	0.88534202
Total (old collection + new collection)	5328	4541	1555	1511	1973
Correctly detected	NA	2973	1399	958	1750
Accuracy	NA	0.654701608	0.899678457	0.634017207	0.886974151