Technical documentation

Contents

1 Lumen Geoguesser

1.1 Notices:

Although you might be reading this documentation in the form of a PDF file, **we highly recommand that you open the README.md file in a markdown editor** (GitHub, VSCode, PyCharm, IDE...). As for the API documentation, after setting up the environment, we recommand you run the server with the `python3 src/app/main.py` command after which you can inspect API endpoints in browser (and execute them too!). Essentialy, the techincal documentation PDF is rendered from the README.md markdown file and export of the in-browser API documentation.

Few more notes:

- the documentation assumes you are located at the `.lumen-geoguesser` directory when running Python scripts
- all global variables are defined in `src/config.py` and `src/paths.py`
- other directories have their own `README.md` files which are hopefully
- you can run most python files with the `python3 program.py -h` to the sense of which arguments you can/must send and what the script actually does

1.2 Directory structure

Directory	Description
data	dataset, csvs, country shapefiles
models	model checkpoints, model metadata
references	research papers and competition guidelines
reports	model stat's, figures
src	python source code

1.3 Setup

1.3.1 Virtual environment

Create and populate the virtual environment. Simply put, the virtual environment allows you to install Python packages only for this project (which you can easily delete later). This way, we won't clutter your global Python packages.

```
[ ! -d "venv" ] && (echo "Creating python3 virtual environment"; python3 -m venv venv)

pip install -r requirements.txt
```

1.3.2 Dataset setup

The original dataset strucutre has a directory `data` with images and `data.csv` at the top level:

```
dataset_original_subset/
├── images
│   ├── 6bde8efe-a565-4f05-8c60-ae2ffb32ee9b
│   │   ├── 0.jpg
│   │   ├── 180.jpg
│   │   ├── 270.jpg
│   │   └── 90.jpg
│   ├── 6c0ed2ea-b31b-4cfd-9828-4aec22bc0b37
│   │   ├── 0.jpg
│   │   ├── 180.jpg
│   │   ...
│   ...
└── data.csv


dataset_external_subset/
├── images
│   ├── e61b6e5f-db0d-4f57-bbe3-4d31f16c5bc3
│   │   ├── 0.jpg
│   │   ├── 180.jpg
│   │   ...
│   ...
└── data.csv
```

Before running other scripts you have to properly setup new dataset structure using the `src/preprocess_setup_datasets.py` file. It's important to note that this file accepts multiple dataset directories as an argument and it will make sure to merge the datasets correctly. No changes will be done to your original directories.

```
python3 src/preprocess_setup_datasets.py -h

usage: preprocess_setup_datasets.py [-h] [--dataset-dirs dir [dir ...]] [--out-dir dir] [--copy-images] [--spacing SPACING

optional arguments:
  -h, --help            show this help message and exit
  --dataset-dirs dir [dir ...]
                        Dataset root directories that will be transformed into a single dataset
  --out-dir dir         Directory where compelte dataset will be placed
  --copy-images         Copy images from dataset directories to the new complete directory.
                        You don't need to do this as later on you will be able to pass multiple dataset directories to var
  --spacing SPACING
                        Spacing that will be used to create a grid of polygons.
                        Different spacings produce different number of classes
                        0.7 spacing => ~31 classes
                        0.5 spacing => ~55 classes
                        0.4 spacing => ~75 classes
                        0.3 spacing => ~115 classes
```

Example of running the initial setup script:

```
python3 src/preprocess_setup_datasets.py --dataset-dirs data/dataset_original_subset data/dataset_external_subset --out-di
```

What this script does on a high level: 1. For all data directories, split the dataset into train, val and test directories 2. `complete_subset/data.csv` is csv has concaternated rows of all `data.csv` s from data directories 3. *Rich static CSV* contains region information, which locations (images) are valid etc, centroids... 4. You can also copy images from all dataset directories to the `dataset_complete_subset` with '– have also

New dataset structure:

```
dataset_complete_subset/
├── data.csv
```

```
└── data_rich_static__spacing_0.5_classes_55.csv


dataset_original_subset/
├── data.csv
├── images [100 entries exceeds filelimit, not opening dir]
├── test
│   ├── c4a74f0d-7f30-4966-9b92-f63279139d68
│   │   ├── 0.jpg
│   │   ├── 180.jpg
│   │   ...
│   ...
├── train
└── val


dataset_external_subset/
├── data.csv
├── images
├── test
├── train
└── val
```

1.3.3 Training

After you prepared that new dataset structure you can start the *quick version* of training

```
python3 src/train.py --dataset-dirs data/dataset_external_subset/ data/dataset_original_subset/ \
--csv-rich-static data/dataset_complete_subset/data_rich_static__spacing_0.7_classes_31.csv \
--quick
```

1.3.4 I have the directory `images` that looks like this: Creating enriched dataframe with centroids and regions:

1.4 Evaluate:

```
curl -X POST lumen.photomath.net/evaluate \
-F 'file=@mapped_to_country_pred-Mike_41-2022-05-06-10-01-15.csv' \
-F "team_code=<INSERT CODE HERE>"
```

Stats: 33.37094934360599 - mapped_to_country_pred-Mike_41-2022-05-06-10-01-15.csv

1.4.1 Developer notes:

To create `requirements.txt` use the following steps:

```
pip install pipreqs
cp requirements.txt requirements.txt.backup
pipreqs --force .
```

```
run python3 src/train.py --accelerator gpu --devices 1 --num-workers 32 --batch-size 8 --dataset-dir data/raw/ data/external/ -
-cached-df  data/complete/data_huge_spacing_0.21_num_class_211.csv  --image-size  224  --lr  0.00002  --unfreeze-at-
epoch 1 --scheduler plateau --val_check_interval 0.25 --limit_val_batches 0.4
```

Merging PDFs:

```
pdfunite in-1.pdf in-2.pdf in-n.pdf out.pdf
```