Maintenance Manual

Development

The technology stack used is Python based, and the environment used to manage packages was the Conda virtual environment. All development for this project was done using an M1 MacBook Air, apart from certain Jupyter Notebooks(file extension `.ipynb`), which were exported into Google Colab and ran. This is because they required more compute.

Conda environment

The Conda environment used is called `hons`. The `environment.yml` file acts as a single source of truth. To replicate the development environment:

1. Clone this repository.
2. `cd` into the `honours_project` directory
3. Run the command:

```
conda env create --file environment.yml
```

> Note: To update the local environment after the `environment.yml` file has been updated, use the command `conda env update --file environment.yml`.

Dependencies

- A `.env` file that contains a Hugging Face token(`$HF_TOKEN`). You can create this with the commands `touch .env` and `nano .env` to edit the file if you are on a Unix machine or MacOS. An example `.env` file is shown in `.env.example`. The Hugging Face token can be acquired by going to the HuggingFace tokens page, located here.
- Ollama installed on your machine.
- Conda - which can be run by following the instructions above.
- TailwindCSS is a CSS library with pre-existing styles. It can be downloaded using `npm install tailwindcss @tailwindcss/cli`. After adding new styles to the `webview/templates` html files, you can use the command `npx @tailwindcss/cli -o webview/static/css/tailwindstyles.css` to update the .css file with the new styles. This is done so that there is always a local version of the CSS file inside the repository; allowing the report to work offline.

Tests

To run the builtin tests, run `python3 -m unittest discover`. These tests, among other things, check whether the relevant files are present in the `./datasets` folder.

Directory Structure

The honours_project directory is split up into 7 folders:

- `datasets` which is designed to hold all the .csv files.
- `logs` which keeps the debug logs of all the scripts.
- `manuals` which holds the user manual and the maintenance manual.
- `modelfiles` which holds example Modelfiles designed for use with Ollama.
- `scripts` which holds the HPC scripts.
- `src` which holds the main scripts such as `dataManipulation.py` to get the training dataset for this project from the CT-RATE dataset, `trainingData.ipynb` which takes in the training data and partitions it into the 5 sets,`evaluateData.py` which evaluates the data created by `prelim_eval.ipynb`, `prelim_eval.ipynb` which produces the inferences for the 3 models used in this experiment and `finetuning.ipynb` which finetunes a model.

  Note that the `finetuning.ipynb` notebook requires an Nvidia GPU on your system. This has different requirements so it is best to run it on Google Colab using a Tesla T4 GPU. Whilst the `prelim_eval.ipynb` notebook does not require a GPU, the inference will be faster if it is moved to Google Colab.

- `tests` which are a series of tests written using Python's `unittest` library, that check whether the correct datasets are on your system, inside the `datasets` and `datasets/inference` folders.
- `utils` which houses the schema and the prompts used in this experiment.
- `webview` which houses the radiology tool, written in HTMX and hosted using Flask. The radiology tool can be accessed using `flask run`.

The supplementary files are:

- `.env.example` an example of a .env file which is required for running the Jupyter notebooks. This should be renamed to .env after downloading the repository.
- `environment.yml` which is a list of the Conda packages required to run this project.
- `evaluateData.py` which is the evaluation suite after performing the inference.
- `README.md` which provides an overview of the project.
- `setup.py` which sets up this project. [NOTE DOES NOT WORK CURRENTLY]

All scripts can be run using `python3 <script-name. >` For example, to run the dataManipulation script, you use `python3 src/dataManipulation.py`.