Reproducibility guide

Team semester 5

June, 2024

Abstract

This document contains a brief introduction on how to maintain reproducibility of the projects semester 5.

Contents

Reproducibility guide

Code and environment

1. **Code Versioning**: Use a version control system like Git to track changes and maintain versions of the code. Tag or create a release of the specific version used for the results.
2. **Environment specification**: Provide a detailed list of dependencies and their versions using a `requirements.txt` file or `environment.yml` for Python environments. Consider using containerization tools like Docker to encapsulate the environment. Provide a Dockerfile or Docker image that can be used to recreate the environment.
3. **Random seed**: Set random seeds for all libraries (e.g., NumPy, TensorFlow, PyTorch) to ensure consistent results across runs. Document the seeds used in the code.
4. **Detailed instructions**: Provide step-by-step instructions for setting up the environment and running the code. This can be included in the README file or a separate setup guide.

Data handling

1. **Data access**: Ensure that the datasets used are publicly accessible. Provide clear instructions on how to obtain and preprocess the data. If the data is modified or preprocessed, include scripts that perform these steps.

2. **Data versioning**: If using a dataset that may change over time, include a snapshot of the data or use a versioned dataset from a repository like Zenodo or DataCite.

Model training and evaluation

1. **Training scripts**: Provide scripts for training the model from scratch. Ensure that these scripts can be executed without modification to reproduce the results.
2. **Pre-trained models**: If providing pre-trained models, include the exact versions and configurations used for training.
3. **Hyperparameters and configurations**: Document all hyperparameters and configurations used for training and evaluating the models. Consider using configuration files (e.g., JSON, YAML) to make these settings easily adjustable.
4. **Cross-validation**: If cross-validation is used, ensure that the code splits the data in the same way each time by fixing the random seed.

Documentation and reporting

1. **Comprehensive documentation**: Ensure that every function and module is well-documented with docstrings explaining their purpose and usage. Include a README file with an overview of the project, setup instructions, and how to run the code.
2. **Results and outputs**: Provide the exact commands or scripts used to produce the results shown in any reports or papers. Include logs, output files, or intermediate results where relevant.
3. **Error analysis and visualizations**: Document the process for generating any visualizations or error analyses, including the code used to produce them.

Additional best practices

1. **Automated testing**: Implement automated tests to verify that the code works as expected. Use frameworks like PyTest or unittest for Python.
2. **Continuous integration**: Set up a CI pipeline (e.g., using GitHub Actions, Travis CI) to automatically run tests and checks whenever changes are made to the codebase.
3. **Reproducibility checklist**: Create a reproducibility checklist that covers all the above aspects. This can be a part of the documentation to help others verify that they have everything needed to reproduce the work.

Example of a README section for reproducibility

Reproducibility

To ensure the reproducibility of our results, please follow the steps below:

Environment setup

1. Clone the repository and navigate to the project directory:

```
git clone https://github.com/yourusername/yourproject.git
cd yourproject
```

2. Set up the Python environment using requirements.txt:

```
pip install -r requirements.txt
```

Alternatively, use the provided environment.yml to create a conda environment:

```
conda env create -f environment.yml
conda activate yourproject
```

(Optional) Use Docker to create an isolated environment:

```
docker build -t yourproject .
docker run -it yourproject
```

Data preparation

Download the dataset from [dataset source] and place it in the `data/` directory. Run the data preprocessing script:

```
python preprocess.py --input data/raw --output data/processed
```

Model training

Train the model using the training script:

```
python train.py --config configs/train_config.yml
```

Ensure that the configs/train_config.yml file contains the exact hyperparameters used in our experiments.

Evaluation

Evaluate the model using the evaluation script:

```
python evaluate.py --model_path models/yourmodel.pth --data_path data/processed
```

The evaluation script will output the results to the `results/` directory.

Reproducibility notes

- All random seeds are fixed in the scripts for consistency.
- Detailed logs are saved in the logs/ directory for reference.
- Please refer to the Dockerfile for creating a reproducible environment using Docker.
- For any issues, refer to the troubleshooting section in `docs/troubleshooting.md`.

T. Busker, Aug. 2025