MeteoSaver v1.0 User Manual

1 Introduction

MeteoSaver v1.0 is a machine-learning-based software designed for the transcription of historical weather data. As a Python-based application, users should ensure they have a compatible Python environment or Jupyter Notebook in order to run it. This manual provides step-by-step instructions on setting up, configuring, and running MeteoSaver v1.0, and also demonstrates how to execute a Minimal Working Example (MWE) in Jupyter Notebook.

2 Installation

2.1 Cloning the Repository

To get started, clone the MeteoSaver repository from GitHub using a terminal or Jupyter Notebook (see command below). This will download all the necessary modules and files for you to successfully run MeteoSaver.

```
!git clone https://github.com/VUB-HYDR/MeteoSaver.git
```

2.2 Setting Up the Environment

After the repository has been downloaded to your local machine or HPC directory, navigate into the MeteoSaver folder to begin setting up the environment:

```
import os

os.chdir("MeteoSaver")
```

MeteoSaver supports two environment setup options: Conda and Docker. Choose your preferred method below and follow the corresponding steps.

2.2.1 Conda Environment

i. Ensure you have Anaconda or Miniconda installed to your system.

ii. Create a Python environment compatible with MeteoSaver using the provided environment file:

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

iii. Following this, the newly created environment:

```
conda activate transcribing_drc_data_environment
```

2.2.2 Docker Environment

i. Ensure you have Docker installed on your system.

ii. Build the Docker image using the Dockerfile in the MeteoSaver directory:

```
docker build -f Dockerfile -t transcribing_drc_data_environment
```

iii. Run the Docker container:

```
docker run -it -v /local_data:/docker_data_dir transcribing_drc_data_environment
```

2.3 Tesseract-OCR Setup (Required)

To ensure the successful execution of MeteoSaver, a final key requirement is the correct setup of Tesseract-OCR. You need to specify the path to the tesseract executable as well as the path to the trained language data.

2.3.1 Setting the Tesseract Path

Depending on your computing environment (personal computer or HPC infrastructure), follow the appropriate steps:

- **Local Setup (Personal Computer)**
  i. Download and install Tesseract from the official installer.
  ii. After installation, locate the path to the Tesseract executable, typically:

  tesseract_path = "C:/Program Files/Tesseract-OCR/tesseract.exe"

- **HPC Environment**
  If you're working on an HPC infrastructure, contact your system administrator to either:
  i. Provide the path to the Tesseract executable, or
  ii. Load the Tesseract module if it's already available.

  Example path for an HPC setup:

  tesseract_path = "/apps/brussel/RL8/skylake-ib/software/tesseract/5.3.4-GCCcore-12.3.0/bin/tesseract"

2.3.2 Specifying the Language Data Path

Tesseract requires access to a language data directory (tessdata) that contains OCR/HTR models.

- **Local Setup (Personal Computer)**
  i. Locate the system tessdata directory. For example:
  system_tessdata_dir = "C:/Program Files/Tesseract-OCR/tessdata"

  ii. From the MeteoSaver repository, copy the custom trained language file located at:
  /OCR_HTR_models/cobecore-V6.traineddata

  iii. Paste this file into your system's tessdata directory, e.g.:
  C:/Program Files/Tesseract-OCR/tessdata/

- **HPC Environment**
  In your job submission script (e.g., **job_script.sh**, also available in the MeteoSaver directory), export the custom tessdata path as an environment variable:

```
export TESSDATA_PREFIX="/OCR_HTR_models/"
```

3 Configuration

Before running MeteoSaver, update **configuration.ini** located in its directory with your user-specific necessary settings:

- **General:** Specify execution environment i.e. ***local*** (Sequential processing on a personal computer) or ***hpc*** (Parallel processing using multiple processors, suitable for High Performance Computing (HPC) environments). This is set to ***local*** by default
- **Directories:** Here, you specify your user-specific directories for the following:
    i. **full_datadir**: Directory containing historical weather datasheet images, organized in folders per station. Not that these images within these folders (stations) have to be named following a specific naming convention: "STN_YYYYMM_SF" or "STN_YYYYMM_HD", based on the data inventory. Here, STN refers to the three-digit station number, YYYY is the year, MM is the month, SF represents Standard Format (printed tabular format), and HD indicates a hand-drawn version of the standard format.
    ii. **pre_QA_QC_transcribed_hydroclimate_data_dir**: Directory for pre-QA/QC transcribed data
    iii. **post_QA_QC_transcribed_hydroclimate_data_dir**: Directory for post-QA/QC transcribed data
    iv. **validation_dir**: Directory for validation results comparing manually transcribed data with MeteoSaver output
    v. **final_refined_daily_hydroclimate_data_dir**: Directory for final refined (QA/QC-verified) daily hydroclimate data
    vi. **transient_transcription_output_dir**: Directory for intermediate/transient outputs during processing
    vii. **manually_transcribed_data_dir**: Directory containing manually transcribed data (used for validation)
- **Table and Cell Detection:** Specify user-defined parameters such as the expected number of rows and columns in the data tables.
- **Transcription:** Define OCR/HTR model settings, including model path and language
- **QA/QC:** Specify which parts of the table should undergo QA/QC checks (e.g. column names, thresholds, uncertainty margins, etc.).
- **Data Formatting:** Indicate the location of date-related information within the tables. This is used to convert the transcribed data into time series formats such as .xlsx and .tsv (Station Exchange Format).

4 Running MeteoSaver

4.1 Local Execution

Once the environment is set up and **configuration.ini** has been properly configured, you can run MeteoSaver locally by executing the **main.py** script located in the **src** folder of the MeteoSaver directory:

```
python src/main.py
```

4.2 HPC Execution

To run MeteoSaver on an HPC system, use the provided **job_script.sh** file located in the MeteoSaver root directory.

Be sure to edit this bash script to match your HPC system's configuration, including module loads, environment activation, location of your pretrained language model for Tesseract OCR, and file paths as needed.

5 Minimal Working Example (MWE)

You can run MeteoSaver using the sample dataset of 10 climate datasheet images located in the data folder in the root directory, without modifying any configuration settings. Simply follow these steps:

i. **Set up the environment** (as described in **Section 2.2**)
ii. **Configure Tesseract-OCR** (as described in **Section 2.3**)
iii. **Run main.py** (as described in **Section 4**).

A Jupyter Notebook for the Minimal Working Example (MWE) is available in the manual_and_minimal_working_example folder in the MeteoSaver root directory.

6 Modules Overview

MeteoSaver consists of six key modules:

i. **Configuration:** Reads user settings.
ii. **Image Preprocessing:** Enhances input images.
iii. **Table and Cell Detection:** Identifies tabular structures.
iv. **Transcription:** Recognizes (handwritten or typed) text/values in the detected tables.
v. **Quality Assessment & Control (QA/QC):** Validates transcription accuracy.
vi. **Data Formatting & Upload:** Converts data into structured formats.

7 Troubleshooting

- Ensure all module dependencies in python are installed using the **environment.yml** available in the MeteoSaver root directory.
- If encountering OCR/HTR errors, verify the Tesseract-OCR model paths in configuration.ini.
- For HPC issues, check resource allocation in job_script.sh as well as loaded modules with respect to your HPC-architecture, and contact your HPC admin, if necessary.

8 Conclusion

MeteoSaver v1.0 automates historical weather data transcription efficiently. This manual guides users through installation, configuration, and execution to ensure smooth operation.