Python Implementation of Bidirectional LSTM for Sequential Data Processing

K. Cerek[1]*, A. Gupta[2], D.A. Dao[1], E. Hadjiloo[1] and J. Grabe[1]

[1]*Institute of Geotechnical Engineering and Construction Management, Hamburg University of Technology (TUHH), Hamburg, Harburger Schloßstr. 36, 21079, Germany*
[2]*Department of Mechanical and Mechatronics Engineering, University of Waterloo, Waterloo, Canada*

This repository contains a Python script for processing and analysing sequential data using a Bidirectional Long Short-Term Memory (LSTM) model. The script is designed to perform various configurations, evaluate model performance, and save results for further analysis. It specifically targets parameter studies conducted on simulated CRS element tests, e.g. (Cerek et al., 2024). The script applies different configurations to train the model and evaluate its performance on test data, including varying data percentages, point skips, number of time steps, and batch sizes.

Installation

To run the script, you need to have Python installed along with the following dependencies. You can install them using pip:

```
pip install pandas numpy tensorflow scikit-learn
```

Usage

1. **Prepare Your Dataset:**
   - Ensure your dataset is in the format expected by the script (i.e., tab-separated text files).
   - Place your dataset files in the `Test_Group` directory.
2. **Run the Script:**
   - Ensure the script is in the same directory as `Test_Group` and `Results` directories.
   - Execute the script:

     ```
     python your_script_name.py
     ```
   - The script will process each file in the `Test_Group` directory, train the model with different configurations, and save the results in the `Results` directory.

Configuration Parameters

The script uses the following configuration parameters:

- `data_known_percentage`: List of percentages of data to be used for training (e.g., `[0.6, 0.7, 0.8]`).
- `point_skip`: List of intervals to skip in the dataset (e.g., `[1, 2, 3]`).
- `n_steps`: List of number of time steps for each input sequence (e.g., `[10, 20, 50]`).
- `n_batches`: List of batch sizes for training (e.g., `[5, 10, 20, 50]`).
- `column_name`: Column name to extract from the dataset (e.g., `"sigma1eff[kN/m²]"`).
- `folder_path`: Directory path for saving results.

---

*Corresponding author. E-mail address: kacper.cerek@tuhh.de, ORDCiD: 0009-0007-3881-2206
Data available at: https://doi.org/10.15480/882.13190

- `n_features`: Number of features in the dataset (default is `1` for univariate data).
- `patience`: Patience for early stopping during training.
- `n_epochs`: Maximum number of epochs for training.

Functions

- `split_sequence(sequence, n_step)`: Splits a sequence into input-output pairs.
- `save_loss_rmse(hist, second_layer_dir)`: Saves training loss and RMSE values to a file.
- `calculate_rmse(actual, predicted)`: Computes the Root Mean Squared Error (RMSE) between actual and predicted values.
- `error_calc(pred, test)`: Calculates the percentage error between predicted and test data.
- `check_for_nan(array, array_name)`: Checks for NaN values in an array.
- `process_file(file)`: Processes each file by applying different configurations and saving results.
- `process_configuration(first_layer_dir, filtered_raw_seq, kp, ps, step, batch)`: Configures and trains the LSTM model.

Results

The script will generate the following outputs for each file processed:

- Training loss and RMSE values over epochs.
- Actual data, predicted training data, and predicted test data.
- Error percentages and detailed logs of training performance.

Results will be saved in the `Results` directory, with subdirectories for each configuration.

License

References

Cerek, K., D. A. Dao, E. Hadjiloo, and J. Grabe (2024). *Dataset of Simulated CRS Tests for Advanced Soil Parameter Identification*. DOI: 10.15480/882.9435.

*Corresponding author. E-mail address: kacper.cerek@tuhh.de, ORDCiD: 0009-0007-3881-2206
Data available at: https://doi.org/10.15480/882.13190