Repository Layout for the Project: "Does the Rich Cousin Support the Poor Sister?"

1 Overview

This repository layout is designed to organize and structure all components of the project efficiently, ensuring reproducibility, modularity, and ease of collaboration. It adheres to best practices for machine learning and NLP research repositories, incorporating directories for code, data, documentation, experiments, and results.

2 Root Directory Structure

```
rich-cousin-poor-sister/
|-- README.md
|-- LICENSE
|-- requirements.txt
|-- .gitignore
|-- data/
|   |-- raw/
|   |-- processed/
|   |-- embeddings/
|   |-- parallel_corpus/
|   |-- evaluation/
|-- src/
|   |-- models/
|   |-- preprocessing/
|   |-- training/
|   |-- distillation/
|   |-- reinforcement_learning/
|   |-- evaluation/
|   |-- utils/
|-- experiments/
|   |-- logs/
|   |-- checkpoints/
|   |-- configs/
|-- results/
|   |-- figures/
|   |-- tables/
|   |-- metrics/
|-- docs/
|   |-- paper/
|   |-- methodology/
|   |-- tutorials/
|-- tests/
|   |-- unit_tests/
|   |-- integration_tests/
```

3 Detailed Description of Each Directory

3.1 README.md

This file provides a comprehensive overview of the project, including:

- Project description
- Installation instructions
- Usage guidelines
- Directory structure explanation
- Links to tutorials and documentation

3.2 LICENSE

Contains the licensing information for the repository. For academic projects, an open-source license such as MIT or Apache 2.0 is recommended.

3.3 requirements.txt

Lists all Python dependencies required to run the project. This file ensures reproducibility by specifying exact package versions.

```
torch==2.0.0
transformers==4.30.0
numpy==1.24.0
scipy==1.10.0
nltk==3.8.0
```

3.4 .gitignore

Specifies files and directories to be ignored by Git, such as temporary files, logs, and large datasets.

```
*.pyc
__pycache__/
data/raw/*
logs/*
checkpoints/*
```

3.5 data/

This directory stores all datasets and related resources:

- **raw/**: Contains unprocessed datasets such as Hindi-Chhattisgarhi parallel corpora and NLLB components.
- **processed/**: Stores preprocessed datasets ready for training (e.g., tokenized text, aligned embeddings).
- **embeddings/**: Contains pre-trained embeddings (e.g., word2vec or multilingual embeddings like mBERT).
- **parallel_corpus/**: Includes Hindi-Chhattisgarhi parallel sentences used for training.
- **evaluation/**: Stores test datasets used for model evaluation.

3.6 src/

This directory contains all source code organized into modular subdirectories:

- **models/**: Implements model architectures (e.g., encoder-decoder models, distillation teacher-student models).
  - base_model.py: Base encoder-decoder architecture.
  - distillation_model.py: Teacher-student distillation model.
  - rl_policy_network.py: RL policy network implementation.
- **preprocessing/**: Scripts for data preprocessing (e.g., tokenization, alignment).
  - tokenizer.py: Tokenization routines for Hindi and Chhattisgarhi.
  - alignment.py: Cross-lingual embedding alignment scripts.
- **training/**: Training scripts for models.
  - train_base_model.py: Training script for the base model.
  - train_distillation.py: Training script for knowledge distillation.
- **distillation/**: Implements confidence-guided distillation techniques.
  - confidence_filtering.py: Filtering mechanism based on teacher model certainty.
  - hint_loss.py: Intermediate-level distillation loss implementation.
- **reinforcement_learning/**: RL-related scripts.
  - rl_environment.py: Defines the RL environment.
  - policy_optimization.py: Implements PPO-based policy optimization.
- **evaluation/**: Evaluation scripts for performance metrics.
  - evaluate_bleu.py: BLEU score calculation.
  - evaluate_meteor.py: METEOR score calculation.
- **utils/**: Utility functions (e.g., logging, checkpoint saving).
  - logger.py: Logging utility.
  - checkpoint_manager.py: Checkpoint saving/loading utility.

3.7 experiments/

This directory organizes experimental setups and results:

- **logs/**: Contains logs generated during training and evaluation runs.
- **checkpoints/**: Stores model checkpoints saved during training.
- **configs/**: Configuration files specifying hyperparameters for experiments.

3.8 results/

Stores outputs from experiments such as figures, tables, and performance metrics:

- **figures/**: Visualization outputs (e.g., loss curves, attention maps).
- **tables/**: Tabular results (e.g., comparison of BLEU scores across methods).
- **metrics/**: Quantitative metrics from evaluation runs (e.g., BLEU/METEOR scores).

3.9 docs/

Contains documentation related to the project:

- **paper/**: Drafts of research papers describing the methodology and findings.
- **methodology/**: Detailed explanations of methods used in the project (e.g., RL-guided distillation).
- **tutorials/**: Step-by-step guides for using the repository.

3.10 tests/

Contains test scripts to ensure code reliability and correctness:

- **unit_tests/**: Unit tests for individual components (e.g., tokenization functions, loss functions).
  - Example test file: test_tokenizer.py.
- **integration_tests/**: Integration tests to verify end-to-end functionality of modules (e.g., training pipeline).

4 Example File Content

4.1 Sample Config File: (experiments/configs/base_model_config.yaml)

```
model:
  type: "encoder-decoder"
  hidden_size: 512
  num_layers: 6

training:
  batch_size: 32
  learning_rate: 0.0001
  epochs: 20

data:
  train_path: "data/processed/train_data.json"
  val_path: "data/evaluation/dev_data.json"
```

4.2 Sample Logging Output: (experiments/logs/training_log.txt)

```
Epoch [1/20], Loss: 3.4521, BLEU Score: 12.34%
Epoch [2/20], Loss: 3.0215, BLEU Score: 15.67%
Epoch [3/20], Loss: 2.7849, BLEU Score: 18.45%
...
```

5 Summary

This repository layout ensures that all aspects of the project are organized systematically while maintaining modularity and scalability. It facilitates collaboration among researchers by providing clear separation between code, data, experiments, results, and documentation. Additionally, it adheres to best practices in machine learning research repositories by including testing frameworks and configuration files for reproducibility.