Intercropping Compatibility Model - Code Explanation
== 1. Reproducibility Setup
----------------------- A fixed random seed (42) is set for Python, NumPy, and PyTorch (both CPU and GPU). This ensures consistent results across runs by controlling random weight initialization, data shuffling, and CUDA backend behavior. 2. Dataset + Tokenizer --------------------- • InterCropDataset: Custom PyTorch Dataset class for handling traits, text, and yield tensors. • SimpleTokenizer: Builds a simple word-level vocabulary and encodes text into integer sequences. • avg_range: Safely parses nutrient values like "125–155" or "140" into numeric ranges. • encode_risk: Converts categorical risk levels (Low, Medium, High) into numerical values. • parse_yield_impact: Parses yield impact values (e.g., "+20%", "5 to 8") into normalized floats. 3. Preprocessing JSON Data ------------------------- • Reads the JSON dataset containing intercropping information. • Extracts crop traits (nutrient requirements, water needs, root depth, etc.). • Normalizes numeric traits using MinMaxScaler. • Encodes textual descriptions with the tokenizer. • Produces three tensors: – Trait features – Encoded text sequences – Yield impact labels 4. Compatibility Model --------------------- The model (CompatibilityModel) integrates both numeric traits and text information: • LSTM: Processes trait features into hidden representations. • Embedding Layer: Encodes text sequences. • Transformer Encoder: Enhances representations with attention mechanisms. • MLP: Fully connected layers output two predictions: – Classification logit (compatibility prediction) – Regression value (yield impact prediction) 5. Training Utilities -------------------- • train_model_kfold: Performs k-fold cross-validation training and validation. – Uses two loss functions: * MSELoss for yield regression * BCEWithLogitsLoss for classification – Combines losses with a weighting factor (cls_weight). – Tracks training/validation loss and accuracy. – Saves accuracy and loss graphs per fold. • find_best_lr: Trains with different learning rates and picks the one with the lowest average validation loss. 6. Main Training Loop
-------------------- • Loads dataset and tokenizer from JSON file. • Performs learning rate search with k-fold validation. • Trains the final model with the best learning rate. • Saves the trained model (.pt file) and tokenizer (.pkl file). 7. Outputs ---------- • Model checkpoints • Tokenizer file • Training graphs (loss and accuracy curves per fold) • Final training logs with accuracy and loss values