Intel® Unnati Industrial Training 2025

Problem Statement 1 – Bug Detection and Fixing

Project Overview:

This project implements an automated bug detection and fixing system using a Machine Learning model. It leverages NLP techniques with the CodeBERT model to analyze code snippets, detect potential bugs, and provide corrections.

Technologies Used:

- ✓ **Python**: Core programming language
- ✓ **Transformers (Hugging Face)**: Pretrained model and tokenization
- ✓ **Torch (PyTorch)**: Deep learning framework
- ✓ **Scikit-learn**: Data processing and evaluation
- ✓ **Flake8**: Code linting tool
- ✓ **Pandas**: Data handling and CSV file operations
- ✓ **Datasets (Hugging Face)**: Dataset handling for training

Steps to Run the Project

i. Setup Virtual Environment
ii. Install Required Libraries

Machine Learning Model Used

Model: CodeBERT (RoBERTa-based model by Microsoft)

About CodeBERT -

CodeBERT is a transformer-based model trained on large-scale code repositories to understand programming languages. It is particularly effective for:

- Code classification
- Bug detection
- Code summarization

- Code completion

Implementation Details : -

1. Data Preparation

i. A dataset containing labeled text samples (Positive, Negative, Neutral) is created and stored in a CSV file (train_dataset.csv).
ii. The text samples are mapped to numerical labels for model training.

2. Tokenization

i. The RobertaTokenizer from Hugging Face is used to preprocess code snippets before feeding them into the model.

3. Model Training

i. The dataset is loaded into a Hugging Face Dataset and tokenized.
ii. The RobertaForSequenceClassification model is fine-tuned with the dataset.
iii. The model is trained using the Trainer class with defined hyperparameters (batch size, epochs, etc.).

4. Bug Detection

i. The trained model is used to classify code snippets as containing a bug or not.
ii. Code snippets are tokenized and passed through the model to predict a label.

5. Bug Fixing (Simplified Approach)

i. A placeholder function replaces occurrences of the word "bug" with "fix" (as an example).
ii. Future improvements could involve code generation techniques for actual fixes.

Then, finally run the Detection and Fixing Process

Expected Output

- If a bug is detected, the script prints "Bug detected!" and attempts to generate a corrected version.
- If no bug is found, it prints "No bug detected."

Future Enhancements

- Improve bug-fixing logic using a transformer-based code generation model.
- Enhance dataset with more labeled buggy and fixed code examples.
- Deploy as an API for real-time bug detection and fixing.

Reported by –

Sneha Das

Student code: BWU/BTA/22/167