Bug Detection and Fixing Model using FastAPI

Overview

This project implements a **Bug Detection and Fixing Model** using **FastAPI** for deployment and **LinearSVC** (Support Vector Machine) for bug classification. It detects bugs in code snippets, suggests fixes using regex-based patterns, and provides a simple web interface. You can access the model remotely using **LocalTunnel** or **Ngrok**.

Features

- **Bug Detection**: Detects whether a given piece of code is buggy or not.
- **Automatic Fix Suggestions**: Applies simple regex-based fixes to known issues.
- **Web Interface**: Built-in HTML form for user-friendly code analysis.
- **REST API**: JSON-based endpoints for programmatic interaction.
- **Remote Access**: Tunnel support using **LocalTunnel** or **Ngrok**.
- **Model Persistence**: Saves and loads trained model and vectorizer with Joblib.

Dataset

- Dataset: google/code_x_glue_cc_clone_detection_big_clone_bench (via Hugging Face Datasets)
- Data Fields Used:
    - func1 and func2: Code pairs for clone detection
    - label: Indicates whether code pairs are similar or not (used for binary classification)
- Preprocessing:
    - Concatenation of func1 and func2
    - Transformed using CountVectorizer

Model

- **Algorithm**: Linear Support Vector Classification (LinearSVC)
- **Vectorization**: CountVectorizer with max 10,000 features
- **Training**:
    - Trained on 10,000 code pairs (or less depending on dataset size)
    - 80/20 train-test split
- **Storage**:

- Trained model saved as bug_fix_model.pkl
- Vectorizer saved as vectorizer.pkl

- **Prediction**:
  - Output: Buggy Code or Fixed Code
  - Confidence score using decision_function

Installation

bash

CopyEdit

Clone the repository

git clone https://github.com/your-username/bug-fixer-fastapi.git

cd bug-fixer-fastapi

Install dependencies

pip install -r requirements.txt

Running the Server

bash

CopyEdit

Launch the server

python your_script_name.py

This will:

- Load or train the model
- Start FastAPI server at http://localhost:8000
- Setup public access via **LocalTunnel**

API Endpoints

GET /

Returns a simple HTML form to enter code and get predictions.

POST /predict

Analyze buggy code and get predictions.

Request Body:

json

CopyEdit

```
{
  "code": "def sum(a, b) return a + b"
}
```

Response:

json

CopyEdit

```
{
  "prediction": "Buggy Code",
  "confidence": 0.983,
  "has_error": true,
  "fixed_code": "def sum(a, b): return a + b",
}
```

Evaluation

- **Accuracy Metric**: Accuracy on validation set printed after training
- **Current Accuracy**: Displayed in the console (e.g., 78.9%)
- **Fixing**: Regex-based heuristic fixing (extendable)

Public Access (Tunnel)

Option 1: LocalTunnel (default)

bash

CopyEdit

```
npx localtunnel --port 8000
```

Outputs a public URL like:

arduino

CopyEdit

```
https://random-id.loca.lt
```

Option 2: Ngrok (uncomment in code)

bash

CopyEdit

ngrok http 8000

HTML Web Interface

Accessible at http://localhost:8000 or via tunnel URL.

Features:

- Code textarea input
- Result display with confidence and fix suggestion
- Copy-to-clipboard button for fixed code

Known Issues & Future Work

- **Add More Fixing Patterns**: Current regex rules are limited; extend for better coverage.
- **Enhance UI**: Improve web form with syntax highlighting.
- **Switch to Transformer Model**: Upgrade to T5 or CodeBERT for smarter bug fixing.
- **Advanced Metrics**: Consider using BLEU/CodeBLEU for better evaluation.
- **Use Better Dataset**: Train on a dedicated bug-fix dataset like Krish-05/bug-detection-and-fixing.