Bug Detection and Automated Fixing

Authors:
Thomas A, Soorya K

Karunya Institute of Technology and Sciences

Index

1. Introduction

This research project, 'Bug Detection and Automated Fixing,' aims to develop a machine learning-driven framework to identify and rectify buggy code. By leveraging real-world buggy code repositories and advanced deep learning models, the project seeks to enhance software debugging efficiency. The methodology includes collecting buggy and fixed code pairs, refining the dataset, training robust machine learning models, and deploying an automated system capable of suggesting optimized bug fixes.

2. Data Acquisition

The dataset was sourced from **BugsInPy**, a repository comprising real-world Python programming errors along with corresponding corrections. The extracted dataset was systematically structured into a CSV format for seamless processing.

Repository Link:
BugsInPy - GitHub

A custom script was developed to extract buggy and fixed code segments from the BugsInPy repository:

```
import os
import pandas as pd

def extract_buggy_fixed_code(patch_lines):
    buggy_code = []
    fixed_code = []
    for line in patch_lines:
        if line.startswith("- ") and not line.startswith("---"):
            buggy_code.append(line[2:].strip())
        elif line.startswith("+ ") and not line.startswith("+++"):
            fixed_code.append(line[2:].strip())
    return "\n".join(buggy_code), "\n".join(fixed_code)

def process_bug_patches(dataset_path):
    bug_data = []
    for project in os.listdir(dataset_path):
        project_path = os.path.join(dataset_path, project)
        bugs_path = os.path.join(project_path, "bugs")
        if os.path.exists(bugs_path):
            for bug_id in os.listdir(bugs_path):
                bug_folder = os.path.join(bugs_path, bug_id)
                patch_file = os.path.join(bug_folder, "bug_patch.txt")
                if os.path.exists(patch_file):
                    with open(patch_file, "r", encoding="utf-8") as f:
                        patch_content = f.readlines()
                    buggy_code, fixed_code = extract_buggy_fixed_code(patch_content)
                    bug_data.append([project, bug_id, buggy_code, fixed_code])
    df = pd.DataFrame(bug_data, columns=["Project", "Bug_ID", "Buggy_Code", "Fixed_Code"])
    return df

dataset_path = r"C:\Users\12a13\PycharmProjects\BUGGY\BugsInPy"
df_final = process_bug_patches(dataset_path)
df_final.to_csv(os.path.join(dataset_path, "final_fixed_bug_dataset.csv"), index=False)
```

3. Data Preprocessing

The extracted dataset was filtered and balanced using the following script:

```
import pandas as pd
from sklearn.utils import resample

df = pd.read_csv("final_fixed_bug_dataset.csv")
min_count = df['Project'].value_counts().min()
balanced_df = pd.concat([
   resample(df[df['Project'] == project], replace=True, n_samples=min_count, random_state=42)
   for project in df['Project'].unique()
])
balanced_df = balanced_df.sample(frac=1, random_state=42).reset_index(drop=True)
balanced_df.to_csv("balanced_bug_dataset.csv", index=False)
```

4. Model Development and Deployment

To enhance bug detection accuracy, we implemented two distinct approaches:

Model 1: TF-IDF and Random Forest Classification

- Converts code snippets into numerical representations using **TF-IDF vectorization**.
- Utilizes a **RandomForestClassifier** to categorize buggy code instances.
- Employs **cosine similarity** for automated fix suggestions.

Model 2: CodeT5 for Intelligent Code Repair

- Implements **Hugging Face's CodeT5 model** for generating context-aware bug fixes.
- Fine-tunes using the **transformers** library to optimize prediction quality.

5. Experimental Analysis

Performance of Model 1: The RandomForestClassifier exhibited high precision in detecting buggy code patterns and retrieving optimal fixes from the dataset.

Performance of Model 2: The CodeT5 model demonstrated superior contextual understanding, producing highly refined bug fixes with greater semantic accuracy.

6. Conclusion

By integrating machine learning and natural language processing techniques, this project successfully automates bug detection and correction. Future enhancements may include expanding dataset diversity and refining model architectures for improved efficiency.

7. References

- [BugsInPy Repository] (**https://github.com/soarsmu/BugsInPy**)