Reproducibility Guide (Replication.pdf)

Reproducibility Guide for Bug Reports Analysis

1. Introduction

The guide presents full instructions to duplicate the analysis of GitHub bug reports based on merged_bug_reports.csv dataset contents. The document provides descriptions about the data alongside setup and execution procedures.

2. Dataset Overview

The dataset merged_bug_reports.csv consists of bug reports extracted from GitHub repositories. It includes the following columns:

- **Repository**: The name of the repository where the bug report was submitted.
- **Number**: Unique identifier for the bug report.
- **State**: The status of the report (e.g., open, closed).
- **Title**: Title of the bug report.
- **Body**: Detailed description of the bug.
- **Labels**: Tags associated with the report (e.g., bug, feature request).
- **Comment**: Extracted comments related to the bug report.
- **Codes**: Code snippets included in the report.
- **Command**: Commands mentioned within the report.
- **Class**: Classification of the report (e.g., 1 for bugs, 0 for non-bug-related reports).
- **Related**: Indicator of whether the bug report is related to others (binary value: 0 or 1).

3. System Requirements

To reproduce the analysis, ensure you have the following:

3.1 Hardware Requirements

- Minimum 8GB RAM (16GB recommended)
- At least 10GB of free disk space
- Processor: Intel i5 (or equivalent) and above

3.2 Software Requirements

- **Operating System**: Windows 10/11, macOS, or Linux
- **Python Version**: 3.8 or higher
- **Jupyter Notebook**: Latest version
- **Dependencies**:
    - Pandas
    - NumPy
    - Matplotlib
    - Seaborn
    - Scikit-learn
    - TensorFlow (if applicable)

Install dependencies using:

pip install pandas numpy matplotlib seaborn scikit-learn tensorflow

4. Steps to Reproduce

Step 1: Load the Dataset

1. Open Jupyter Notebook.
2. Load the dataset with:
3. import pandas as pd
4. df = pd.read_csv("merged_bug_reports.csv")
5. df.head()

Step 2: Data Preprocessing

- Check for missing values:
- df.isnull().sum()
- Drop or impute missing values as needed.
- Convert categorical data into numerical if required:
- df['State'] = df['State'].map({'open': 1, 'closed': 0})

Step 3: Exploratory Data Analysis (EDA)

- Count of open vs. closed issues:
- import seaborn as sns
- sns.countplot(data=df, x='State')
- Most common labels:
- df['Labels'].value_counts().head(10)

Step 4: Bug Classification Model (Optional)

- If using machine learning for bug classification:
- from sklearn.model_selection import train_test_split
- from sklearn.ensemble import RandomForestClassifier
-
- X = df[['Number', 'State', 'Class', 'Related']]
- y = df['Class']
- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
-
- model = RandomForestClassifier(n_estimators=100)
- model.fit(X_train, y_train)
- print("Accuracy:", model.score(X_test, y_test))

Step 5: Visualization of Results

- Generate a correlation heatmap:
- import matplotlib.pyplot as plt
- import seaborn as sns
-
- plt.figure(figsize=(10,6))
- sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
- plt.show()

Step 6: Save and Export Processed Data

- Save the cleaned dataset:
- df.to_csv("processed_bug_reports.csv", index=False)

5. Conclusion

The guide establishes conditions for achieving consistent results in the bug report evaluation process. Strictly follow the provided procedure while verifying that all required dependencies get installed correctly alongside validating dataset loading. Changes in results stem from the differences in dataset versions combined with variations in software execution environments.