Policing Bot Software Development Test Plan

Cody Manning
Gabriel Silva
Liam Dumbell

September, 2023

Table of Contents

1. Introduction

1.1 Scope

This test plan concerns the Policing Bot academic senior design project. This document will keep track of the manner in which the software is tested along with how we categorize the test we perform. This document is a guide to the reader to ensure they understand how the software is tested, along with how errors are tracked and checked for. This document is subject to change as the project moves forward and may be updated, any updates within this document will be documented in the final chapter.

1.2 References

1.2.1. 829-2008 - IEEE Standard for Software and System Test Documentation: Link
1.2.2. Policing Bot Software Requirements Specification: Link

1.3 Software Overview

The developed framework will allow for users to detect bots on a schedule or manually. It should be able to detect bots with a predefined accuracy rating (80%). When a bot is detected, it should be able to distinguish between a bot that is deemed as beneficial or malicious. When the bot is detected, it should annotate it as beneficial or malicious and move on to the next stage. The final stage is the decision stage, where the framework should decide what happens to the detected bot, whether it is to be reported for deletion or left alone (but marked for future analysis).

2. Testing Strategy

2.1 Testing Approach

The testing approach for the Twitter Bot Detection Framework will encompass a combination of manual and automated testing. Manual testing will be used for exploratory testing, user interface testing, and certain functional tests, while automated testing will be employed for repetitive and regression testing tasks. Continuous integration and continuous testing practices will ensure that automated tests are executed regularly.

The testing approach for the Twitter Bot Detection Framework will be tailored to address the specific functions of the framework, including:

- Detect Function: To verify the framework's ability to distinguish normal users from bots, the testing approach will primarily involve automated testing using labeled datasets of normal and bot accounts. The framework's accuracy in detecting bots will be assessed against a ground truth.
- Distinguish Function: To evaluate the framework's capability to distinguish between beneficial bots and malicious ones, a combination of manual and automated testing will be used. Testers will manually analyze results to assess the framework's accuracy in categorizing bots.
- Decide Function: To test the framework's decision-making process, automated test cases will be designed to simulate various scenarios where bot accounts are detected and categorized. The framework's action recommendations will be evaluated against predefined criteria.

2.2 Testing Levels

- Unit Testing: Individual components, functions, and algorithms will be tested in isolation to ensure they function correctly.
- Integration Testing: The interactions and data flow between various components of the framework will be tested to ensure seamless integration.
- System Testing: The entire system will be tested as a whole to evaluate its functionality, performance, and compliance with requirements.

To break this down in terms of the overall primary functions of the software, the following test levels will be executed for each of the three main functions:

- Detect Function:
    - Unit Testing: Individual components responsible for bot detection algorithms will be tested in isolation.

    - Integration Testing: The integration of detection components and their interactions with the data preprocessing module will be tested.
    - System Testing: The entire bot detection process, from data collection to classification, will be evaluated.
- Distinguish Function:
    - Manual Testing: Human testers will assess the framework's ability to categorize bot accounts into beneficial and malicious categories.
    - System Testing: Automated tests will evaluate the overall accuracy of the framework in distinguishing between bot types.
- Decide Function:
    - Unit Testing: Testing individual components responsible for decision-making logic.
    - Integration Testing: Evaluating the integration of the decision-making process into the framework.
    - System Testing: Assessing the overall effectiveness of the framework's decisions in different scenarios.

3. Test Cases

3.1 Detect Function

Test Case 1: Detection Accuracy

- Objective: To verify that the framework correctly identifies bot accounts.
- Preconditions: A dataset containing a mix of normal user and bot accounts.
- Test Steps:
    1. Input the dataset into the framework.
    2. Execute the detection process.
    3. Evaluate the results against ground truth labels.
- Expected Result: The framework accurately detects bot accounts with a defined accuracy threshold.

Test Case 2: Real-Time Detection

- Objective: To test the real-time detection capabilities of the framework.
- Preconditions: The framework is connected to a live Twitter data stream.
- Test Steps:

1. Monitor real-time Twitter data.
2. Trigger the detection process upon new data arrival.
3. Verify that real-time bot accounts are identified.

- Expected Result: The framework detects bot accounts in real-time and responds promptly.

3.2 Distinguish Function

Test Case 3: Categorization Accuracy

- Objective: To assess the framework's ability to categorize bot accounts into beneficial and malicious categories.
- Preconditions: A dataset containing examples of beneficial bots, malicious bots, and normal users.
- Test Steps:
    1. Input the dataset into the framework.
    2. Execute the categorization process.
    3. Evaluate the results against ground truth labels.
- Expected Result: The framework accurately distinguishes between beneficial and malicious bots, maintaining a defined accuracy level.

Test Case 4: Beneficial Bot Identification

- Objective: To ensure the framework correctly identifies beneficial bots.
- Preconditions: A dataset containing examples of beneficial bots and normal users.
- Test Steps:
    1. Input the dataset into the framework.
    2. Execute the categorization process.
    3. Verify that beneficial bots are correctly identified.
- Expected Result: The framework accurately identifies beneficial bots without classifying them as malicious.

3.3 Decide Function

Test Case 5: Decision Accuracy

- Objective: To validate the correctness of the framework's decision-making process.
- Preconditions: Bot accounts have been successfully detected and categorized.
- Test Steps:

1. Input the detected and categorized bot accounts into the framework.
2. Execute the decision-making process.
3. Evaluate the framework's recommended actions against predefined criteria.

- Expected Result: The framework correctly decides whether to report bot accounts to Twitter or mark them for further analysis based on defined criteria.

Test Case 6: Reporting Integrity

- Objective: To ensure that the framework correctly initiates the reporting process to Twitter.
- Preconditions: The framework recommends reporting certain bot accounts to Twitter.
- Test Steps:
    1. Trigger the reporting process for flagged bot accounts.
    2. Verify that the reporting process is initiated correctly.
- Expected Result: The framework successfully initiates the reporting process for malicious bot accounts.