Policing Bot Software Development Requirements Specification

Cody Manning
Gabriel Silva
Liam Dumbell

September, 2023

Table of Contents

1. Introduction

1.1 Purpose:

The purpose of this document is to accurately outline and specify the various requirements we believe our proposed framework will satisfy once fully developed. Additionally, this document will serve as a written agreement between the developers and the client Dr. Shloub as to what the project should aim to achieve. This document can also be used as a general reference during the development process to ensure that the development of the program is incrementally satisfying the requirements listed in this document.

1.2 Scope:

The core goal of this project is to deliver a framework to our client that directly interacts with the social media platform Twitter through the use of “Police Bots” that can be controlled using Python code. This framework will firstly be able to detect bot accounts on Twitter with a high degree of accuracy by collecting, compiling and then analyzing account information on accounts thought to have the possibility of being controlled by a bot. This process will also categorize bots based on their apparent purpose on Twitter. Secondly, this framework will be able to distinguish between what we define as a “malicious bot” and a “beneficial bot” by analyzing the account data and what categories the bot accounts fall under. Lastly, this framework will be able to accurately decide whether a malicious bot should be reported to the administrative authorities at Twitter and recommend that the client should do so.
The working scope of this project involves creating a framework that creates and deploys Police Bots on Twitter that can be controlled using Python scripts. This framework will aim to automatically scan Twitter using these Police Bots on an optimized schedule and compile data on accounts thought to be bots. The framework will then “dynamically analyze” (idk if we have to define dynamic analysis) the compiled account data and determine information on a bot account as well as a level of maliciousness and recommend what actions should be taken by the user / client. This will mainly be beneficial to the user base of Twitter as it will lead to a safer platform where users are less likely to be negatively affected by a malicious bot account, but the

findings of this document will provide insight into bot accounts on other social media platforms beyond Twitter.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
Bot	Short for robot. In this case we refer to a software or script that performs automated tasks.
Beneficial Bot	A bot that performs tasks that are helpful or productive to individuals interacting with it E.g.: Translators, weather reports, or meme generators
Malicious Bot/MalBot	A bot that performs tasks that are harmful, dangerous or illegal. E.g.: Scams, spam, or fraud.
Twitter/X	A social media platform that allows users to post text or media and allow other people to see it. Can be used personally by individuals or companies to spread information.
Tweet	Posts or replies to other people's posts in Twitter/X that are publicly available if allowed by the user.
Thread	People can tweet on other peoples' tweets and make a sequence of connected tweets. An analogy of threads in computer science terms would be a tree.
Root tweet/Base tweet	The first tweet of a thread. Similar to the root node of a tree.
Retweet	A repost of another accounts post that appears on a user's profile
Trend/Trending topics	Twitter/X has a defined section of the website that enables users to see a list of

	tweets or hashtags (explanation below) that are being discussed/used by numerous accounts in the last 24 hours.
DM/Direct Message	Messages sent privately to users.
@	Symbol used to mention other users on Twitter. This symbol is followed by a username to create a hyperlink to the user's profile and notify them. E.g.: @taylorswift13
#	Called 'Hashtag'. Is a label used to organize and categorize content to make it easier to discover for other users.
Dynamic Analysis	Dynamic analysis is the testing and evaluation of an application during runtime. In the context of this project, Dynamic Analysis refers to the analysis of account data right after it is compiled.
AWS	Amazon Web Service (database hosting service)
MySQL	MySQL is a type of SQL database.
GDPR	General Data Protection Regulation: a set of standards for handling user data.
API	Application Programming Interface (A set of rules and protocol that allows different software applications to communicate to each other.)
Tweepy	Python library used to integrate with the Twitter API

1.4 References:

1.4.1. 830-1998 - IEEE Recommended Practice for Software Requirements Specifications: Link

1.5 Document Overview:

The remaining chapters of this document will document the functional and technical requirements of the policing bot project, as well as any planned features or functionality. Chapter 2 will detail the functionality of the project. Chapter 3 will describe the requirements that the project needs in order to be accepted.

2. Functionalities

2.1. Product Perspective

Users of our framework will be able to create a Policing Bot that has the purpose of finding possible bot accounts on Twitter and then compiling and downloading their account data. This will be done by interacting with our framework's GUI which will let users deploy a Policing Bot and manage the bot through the GUI. These bots will incrementally scan through tweets starting at a specific time each day and will report their findings through the GUI. The framework will also allow users to view the findings of the analysis of the account data compiled by the Police Bots as well as download them as files. There are other products out there that allow for users to create Twitter Bots that can do a variety of things, but we want to focus on creating Twitter bots that detect other Twitter bots on a fixed schedule. This is what we consider the most efficient approach to compiling data on bot accounts in terms of computing resources while maintaining the integrity of our framework's findings.

We intend to keep our framework self contained to the system it is running off of. This means that the entirety of the analysis of account data is done locally. The only area that exists outside of the local system in our framework are the deployed Police Bots that scan Twitter on a schedule. We do not intend on making the Policing Bots autonomous or a constantly running tool and consider those approaches to data collection as outside the scope of our project.

In order to begin using our framework, a user would have to firstly download a repository off of our website and subsequently run the required commands to launch the framework on their local machine. After doing this, a user can decide to do any of the mentioned actions and can terminate the framework and all of its processes at any time.

2.2. Product Functions

- Bot creation **-** The framework will allow the users to bind an account to the program which will allow them to run our other bot detection methods and subroutines.

- Bot scheduling **-** After an account has been bound to the program, the user will be able to configure the bot to run after a trigger. Some examples of triggers that can be used are times of the day or whenever a new trending topic comes up.
- Bot discovery **-** The framework will enable the bot to read a tweet or set of tweets and download the account data from accounts that interact with these tweets.
- Bot distinguishing - After reading a tweet and selecting an account to analyze, the framework will analyze the account data associated with the chosen account and reliably predict if the account is run by a bot.
- Data storing - After a bot is detected, the framework will tag that account and collect data to send to a database. This data will be used to further develop our detection methods, and for further analysis of bot behavior.

3. Specific Requirements

3.1. External Interfaces:

- User Interfaces: Most of the framework will be run automatically in the background, but the user will be able to interact with a GUI made in Python (can change this to website/updating repo on git/paper publishing our findings/etc) that displays the account information of identified malicious bot accounts as well as the actions recommended by the framework based on its analysis. The user will also be able to download the reports made by the framework and view its findings in an application like Microsoft Excel.
- Hardware Interfaces: The framework will exclusively interact with PCs / Laptops that have Python installed along with the required libraries.
- Software Interfaces: The framework will mainly interface with the desktop version of the Twitter website, along with the Twitter API. The framework will also potentially interface with a database service such as AWS or a local version of a MySQL database. The minimum requirements to run our framework would be a PC with Python installed.

3.2. Functional Requirements:

3.2.1 Data Collection:

- Twitter API integration: The framework shall implement a mechanism to retrieve Twitter data through the Twitter API, including tweets, user profiles, follower/following relationships and other relevant account data.
- Data Storage: Store collected data in a secure and scalable database for future analysis.

3.2.2 Data Pre-processing:

- Text Pre-processing: The framework shall implement text cleaning, tokenization, and feature extraction techniques to process a text tweet.
- Data Labeling: The framework shall annotate data with labels (bot or not bot) for supervised learning.

3.2.3 Bot Detection:

- Passive Detection: The framework shall scan for bots on a schedule, not in real-time. Real-time functionality may be implemented in the future, but not on initial launch.
- Batch Processing: The framework shall be able to analyze batches of historical data in order to properly detect bots from past bots found.

3.2.4 Bot Distinguishing:

- Judgment: The framework shall be able to distinguish between beneficial and malicious bots according to a predefined set of standards.

3.2.5 Bot Decision Making:

- Decide: The framework shall be able to make a decision on what to do with a found malicious or beneficial bot. Some examples of a decision are reporting the bot for removal from the Twitter website, or to simply record it and monitor it for future analysis.

3.2.6 User Interface:

- Dashboard: The framework shall display a user-friendly dashboard to visualize bot detection results and insights, to schedule new scans or manually send out scans.

3.2.7 Export Findings:

- Download: The GUI for the framework will have the option to download the current findings as a Microsoft Excel spreadsheet file which will contain the relevant account data on all bot accounts and the framework's findings in an organized way.

3.3. Non-functional Requirements:

3.3.1 Performance

- Scalability: The framework shall be able to handle a large volume of Twitter data and users.
- Response Time: The framework shall be able to provide quick responses to account activity.
- Accuracy: The framework's accuracy in bot detection shall meet a predefined standard, aiming for a minimum of 80% accuracy.

3.3.2 Security

- Data Privacy: The framework shall ensure user data privacy compliance with relevant regulations (GDPR).

3.3.3 Maintenance and Support

- Documentation: Maintain comprehensive documentation for setup, usage, and troubleshooting.
- Regular Updates: The program shall be updated regularly with the ever changing Twitter API, or general changes to the architecture of the Twitter website.