Machine Learning Twitter Analysis Tool

(Trendy Name [TBD])

|  |  |
| --- | --- |
| **Team Members:** | Chris Bakopoulos, Eric Benton, Bryce Eller, Alex Sladek, Phil Stephenson, Lucas Vitalos, Rush Weigelt |
| **Professor:** | Dr. Jeff Salvage, Ph. D. |
| **Section #:** | Tuesday, 6-6:30PM |
| **Stakeholder:** | David Stahl, atty |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason** | **Version #** |
| Rush Weigelt | 9/20/19 | Creation of base outline | 0.0 |
| Rush Weigelt | 9/21/19 | Met and took notes with Stakeholder | 0.01 |
| Rush Weigelt | 9/30/19 | Convert to legible Requirements Doc | 0.02 |

**Table of Contents**

1. [**Introduction**](#Introduction)
   1. [Purpose of Document](#PurposeOfDoc)
   2. [Scope of Document](#ScopeOfDoc)
   3. [Overview of Document](#OverviewOfDoc)
2. [**Description**](#Description)

2.1 [Product Perspective](#ProductPerspective)

2.2 [‘Bot’](#GameRules) Detection

2.3 Two-Group Categorizer for ‘Potentially Dangerous’ Users

2.4 Python

1. [**Functional Requirements**](#FunctionalRequirements)

3.1 Bot Detection Application

3.2 ‘Potentially Dangerous’ Post Categorizer Application

1. [**Non-Functional Requirements**](#NonFunctionalRequirements)

4.1 [Network Performance](#NetworkPerformance)

4.2 [Load Management](#LoadManagement)

4.3 [Host Operating System Requirements](#HostingOperatingSystemRequirements)

4.4 [Accessibility](#Accessibility)

1. [**User Interface**](#UserInterface)

5.1 [Checkers Game](#CheckersGame2)

5.2 [Server](#Server2)

5.3 [Main Menu/UI](#MainMenu_UI)

1. [**Use Cases**](#UseCases)

6.1 [Use Case Flow](#UseCaseFlow)

6.2 [Selecting a Checker to Move](#SelectingAChecker)

6.3 [Making and Checking a Valid Move](#MakeAndValidateMove)

6.4 [Update Other Player’s Screen](#UpdateScreens)

6.5 [Start Server](#StartServer)

6.6 [End Server/Game](#EndServer)

1. [Introduction](#ToC)

* 1. Purpose of Document

This document will detail the requirements for our machine learning Twitter analysis tool.

* 1. Scope of Document

This document is scoped to enable a new programmer to join the team and be able to integrate themselves into the team within a day or two. It will lay out how the game will be coded, without ambiguity.

* 1. Overview of Document

Following this introduction will be our functional and non-functional requirements for *[TBD]*.It will also feature use-cases and UI mock-ups.

2. [Description](#ToC)

**2.1 Product Perspective**

*[tbd]* is a python-based application. It runs a machine learning Twitter analysis tool focused on two functions: 1) Detecting and reporting suspected ‘Bot’ accounts, with an attempted focus on politically-oriented bots and 2) Detect and log “potentially dangerous” tweets based on a supervised learning dataset.

**2.2** **‘Bot’ Detection**

‘Bot’ accounts, or automated accounts meant to mimic real users, are a prevalent issue on social media—none moreso than Twitter. While bots are an ever-present issue, analysts predict the run-up to the 2020 US election will see a large influx of fake users.

In an attempt to effectively help counteract the issue, we propose an application that will have two functions: 1) webscrape and, using supervised machine learning categorize, a certain number of tweets for a certain ‘hashtag’. Any posts categorized as “Likely Bot” would be linked and saved to a CSV file. This way, a human could then inspect the suspect accounts before deciding whether to report the account.

**2.3** **‘Potentially Dangerous’ Categorizer**

Develop an online dataset based on the posts and manifestos of convicted mass murderers. Using this dataset, use supervised learning to categorize posts from a provided hashtag into two groups. Export the ‘potentially dangerous’ posts to a CSV, to be inspected by an attorney or intern.

**2.4** **Python**

Python is the primary language we will be working in. This will allow us to utilize a bevvy of different modules.

|  |  |
| --- | --- |
| **Module** | **Reason for Use** |
| Sklearn | Machine Learning |
| Numpy | calculations |
| Pandas | Dataframes |
| Pathlib | Universal paths |
| BeautifulSoup | Webscraping |
| Twitter API | Access to Twitter’s API |
| Mathplot | Basic visualization |

3. [Functional Requirements](#ToC)

**3.1** **‘Bot’ Detection**

An application that has all of the following:

1. User-input for a desired hashtag.
2. Webscrape a given number of the most recent posts.
3. Use supervised machine learning to categorize potential bots.
4. When done, export suspected accounts to a CSV file.

**3.2 ‘Potentially Dangerous’ Categorizer**

An application that has all of the following:

1. User-input for a certain hashtag.
2. Webscrape a given number of the most recent posts.
3. Use supervised machine learning to categorize posts.
4. Include a progress bar to indicate progress.
5. Export concerning accounts to a CSV file.
6. Save plotted ML results as a PNG file.

4. [Non-Functional Requirements](#ToC)

**4.1 ‘Bot’ Detection**

An application that has all of the following:

1. User must have an understandable UI to enter the desired hashtag.
2. Number of webscraped posts must be customizable, but within confines determined by commonsense (for now, greater than 300, less than 100,000).
3. Include an accurate progress bar to represent estimated completion time when categorizing.
4. Export links to accounts as a CSV that is readable in Microsoft Excel.
5. CSV should save to a local file.

**4.2 ‘Potentially Dangerous’ Categorizer**

An application that has all of the following:

1. User must have an understandable UI to enter the desired hashtag.
2. Number of webscraped posts must be customizable, but with confines determined by commonsense (for now, greater than 300, less than 100,000).
3. Must use supervised learning, to avoid claims of human-bias.
4. Include an accurate progress bar.
5. Export links to accounts as a CSV.
6. Save graph of results as a PNG file in a local directory.

5. [User Interface](#ToC)

**5.1 Application**

1. A dropdown menu to select whether it runs the search for bots or potentially dangerous posts.
2. User-supplied hashtag to concentrate on.
3. Dropdown menu to select from a range of recent posts to search through.
4. A ‘Run’ button in the bottom right.
5. An accurate progress bar to estimate time-to-completion.



6. [Use Cases](#ToC)

**6.1 Use Case Flow**

The following Use Cases are given in their likely sequential flow for the User. Differences between the two

**6.2** **Launch Application**

Make application launchable via Microsoft GUI, not command line.

**Precondition:** None

**Action:** User clicks to launch application.

**Postcondition:** The application runs, displaying the UI.

**6.3 Making and Checking a Valid Move**

Valid moves, which will be programmed in via making use of a 2D-Array and splitting the board into quadrants, will be highlighted for the player after a piece is selected. A back arrow will be in the top left corner, if the user wants to deselect the selected piece.

Otherwise, clicking on any highlighted space will make the valid move chosen.

**Precondition:** A piece has been selected, and an open, valid space is highlighted.

**Action:** The user clicks a highlighted, valid space.

**Postcondition:** The piece moves there. If an opponent’s piece is jumped, that is resolved in the following phase.

**6.4 Resolving Jumped Opponent’s Pieces**

If an opponent’s piece was jumped in the previous phase, it is removed after the valid move is completed.

**Precondition:** The opponent’s piece was jumped in the previous phase.

**Action:** The piece is removed.

**Postcondition:** The piece is no longer on the board/usable by the opponent.

**6.5 Update Other Player’s Screen**

After the move and any removed pieces are resolved, the updated board is sent to the opponent on the server.

**Precondition:** The previous board layout.

**Action:** The player’s turn is completed.

**Postcondition:** The corresponding moves appear on the other user’s screen.

**6.6 Check for End Game State**

After both screens are updated, the game will check whether the endgame state has been reached. The endgame state for checkers is the opponent having 0 pieces remaining on the board.

**Precondition:** The board after a given turn.

**Action:** Either player’s turn ending.

**Postcondition:** The user in control switches *or* the game is pronounced over if one player has no pieces left.

**6.6 Start Server**

To start a game, the hosting player will start a server.

**Precondition:** No player has attempted to start a game yet.

**Action:** The first player starts a game.

**Postcondition:** A server is opened, and the host player can invite another player who joins the server.