Software Requirements Specification: Twitter Bad-Actor Detector

Richard Colorusso, Alisha Mishra, Jeremy Thrasher, William Zhu

COMP 4960-07

**Revision History**

| Revision | Date | Description |
| --- | --- | --- |
| 01 | 02/03 | Initial Draft |
| 02 | 02/09 | Update Requirements |
| 03 | 02/10 | Update Requirements, Edit for clarity |
| 04 | 02/12 | Edit for clarity and detail |
| 05 | 02/16 | Edit for clarity, fix language |
| 06 | 02/17 | Edit for vocabulary, improved detail |
| 07 | 02/24 | Edit for vocabulary, improving detail, expanding requirements |

**1. Purpose**

The purpose of this document is to outline the software requirements for an application to detect Twitter Bad Actors. The expected user audience for this application is general Twitter users looking to run credibility checks on accounts posting information that are easily misdirected, such as political or medical topics. The audience for this document includes developers, product or project managers, and general users.

**2. Scope**

The application uses various features such as n-gram, network analysis, intent/sentiment analysis to predict the chances of an account being a bad actor. It also highlights specific tweets that are the most significant in the said analysis.

**3. Definitions**

*Bad-Actor*: A user on a service with the goal of harassing people or spreading misinformation

Stochastic Analysis: Making predictions about a subject based on statistical likelihoods instead of definitive truths

*N-gram*: A sequence of n consecutive words. For example, in the phrase “I am a robot”, possible bi- (2-) grams are “I am”, “am a”, and “a robot”.

*API*: Application Programming Interface, or standard defined for two computer applications to communicate with each other

*JSON*: Javascript Object Notation, which is a standard format to communicate with APIs

*View Pooling*: A technique in neural networks in which the results of multiple networks are used to generate a single result

**4. Introduction**

4.1 Problem Statement

Misinformation and automated information campaigns represent a significant threat to the flow of information on social media. Bad actors increase the reach of scams, spread Covid-19 misinformation, and influence elections. The environment created by these accounts then leads to regular users having less faith in the platform itself and other users.

4.2 Proposed Solution

This product is a website and API that generates a score representing how likely an account is to be a bad-actor. A user suspicious about another account can look up that account’s score, and then either stop interacting with a user with a high score. Users will also have greater trust in a user with a low score. This will enhance people’s overall twitter experience since they can have a greater background information when interacting with new accounts and identify bad-actors to ignore.

4.3 Novelty

Specific detection for bad-actors is difficult and relies on stochastic analysis rather than absolute indicators. Twitter already tags some forms of bad-actor accounts like misinformation around elections or COVID-19 and there is another application called Block Party that tags users based on hateful content like slurs and hate speech along with account metadata, but our product is both uniquely holistic in how it generates a score and uniquely transparent in how it displays that score to the end user. By offering this additional functionality, we will inspire greater trust in our platform and let users navigate through twitter more securely.

4.4 User Characteristics

A typical user for this product does not have a specific range of technical knowledge and specific knowledge of machine learning. No specialized knowledge is required to use the product.

4.5 Product Functions

The application includes the following functions for users:

* Ability to input any user into the search bar
* Utilize a drop down menu list for fast user selection
* View bad-actor report and score
* View highlighted tweets directly influencing the given score
* View a general description for users to see why an account received the score it did

The application includes a simple, fast interface for a user to conduct a speedy review on an account. A user of the application will not have to sift through significant amounts of information and text, they will be able to get the data they are looking for in less than a minute.

**5.1 Functional Requirements**

**5.1.1 Website**

5.1.1.1 There shall be a search bar.

5.1.1.1.1 The Search Bar shall accept as input any alphanumeric character (a-z,A-Z,0-9) as well as underscores.

5.1.1.1.1 The Search Bar shall display an error message when an invalid character is input when searching for an account.

5.1.1.1.2 The Search Bar shall display the message “No Account Found” dialogue when no user is found that matches the input characters

5.1.1.1.3 The system shall permit the user to enter the name of any valid twitter account.

5.1.1.1.4 The search bar shall display a dropdown list of users which start with the characters typed in the search bar.

5.1.1.2 There shall be a webpage titled ‘Score and Highlights’.

5.1.1.2.1 The webpage will display the following characteristics of the selected account:

* Username
* Profile picture
* Account bio
* Follower count.

5.1.1.2.2 This page shall display a number labeled ‘Overall Account Score’.

5.1.1.2.3 This number shall be an integer between 0 and 99.

5.1.1.2.3 This page shall include a visual model related to the score generated.

5.1.1.2.4 This page shall display text explaining the analysis.

5.1.1.2.5 The text analysis shall be in English.

5.1.1.2.6 The text analysis shall be either 3 or 4 sentences long.

**5.1.2 Score Generation**

5.1.2.1 The system shall contain a service called “N-gram Analysis”.

5.1.2.1.1 This service shall include a file with 1000 phrases used by bad actors.

5.1.2.1.2 There shall be a web service that accepts a string representing a username as input and returns a value representing a score.

5.1.2.1.3 This score shall be a floating-point value in [0,1).

5.1.2.2 Account Metadata Score

5.1.2.2.1 There shall be a web service that generates a prediction based on the following criteria:

* Account metadata
  + Geolocation
  + Account date creation
  + Tweet frequency
  + Follower count

5.1.2.2.2 This web service shall return a floating-point value in [0,1).

5.1.2.3 Sentiment Analysis

5.1.2.3.1 There shall be a web service that generates a score of how hostile a user’s tweets are.

5.1.2.3.2 This web service shall return a floating-point value in [0,1).

5.1.2.4 Score Generator

5.1.2.4.1 There shall be a web service that generates an overall score for a given account.

5.1.2.4.2 The web service shall return an integer value in [0,99].

**5.1.3 Database**

5.1.3.1 Database shall hold the following three parameters used for score determination.

* n-gram analysis
* metadata
* intent analysis

5.1.3.1.1 The database shall contain a mechanism to add additional features and parameters without deleting existing data in the database.

5.1.3.1.2 The database shall be scalable.

**5.1.4 API**

5.1.4.1 POST Request

5.1.4.1.1 This API shall accept a POST request in the following JSON format and update the database automatically based on the contents of that request.

{“bearer\_token”:string,

“username”:string,

“overall\_score”:float,

“ngram\_score”:float,

“metadata\_score”:float,

“sentiment\_score”:float }

5.1.4.1.2 This API shall require a bearer token which is non-anonymous.

5.1.4.2 GET Request

5.1.4.2.1 This API shall accept a GET request in JSON format and return a JSON file.

Input Format:

{“bearer\_token”:string

“username”:string }

Output Format:

{“username”:string,

“overall\_score”:float,

“ngram\_score”:float,

“metadata\_score”:float,

“sentiment\_score”:float }

5.1.4.2.2 This API shall require a bearer token which is non-anonymous.

**5.2 Non-Functional Requirements**

5.2.1 Efficiency: This system shall sequentially do the following things without noticeable latency.

* accept user input
* process data
* display results

5.2.1.1 Speed: Querying a user shall take no longer than 10 seconds for 99.9% of lookups.

5.2.1.2 Performance: This program shall supply accurate data upon request without failing.

5.2.2 Ethics: The program shall include a statement for the user regarding ethics and data. privacy

5.2.3 UI Elements

5.2.3.1 Welcome Page

5.2.3.1.1 The webpage shall contain a header and navigation bar at the top.

5.2.3.1.2 The webpage shall contain a search bar centered in the middle of the screen.

5.2.3.1.3 The webpage shall contain a mission statement below the search bar.

5.2.5.1.4 The webpage shall contain a message citing us as authors.

5.2.3.2 Rating Page

5.2.3.2.1 The rating shall include a navigation bar and a header.

5.2.3.2.2 The rating page shall display the name and profile picture of the searched profile at the top.

5.2.3.2.3 The rating page shall display the score generated by the backend.

5.2.3.2.4 The page shall list the features that were most significant in determining the rating.

5.2.3.2.5 If the user’s rating is above 50, the page shall list 3 posts that were most significant in determining the rating.

**References**

Ian Sommerville. Software Engineering, 10th ed. Pearson, 2015. ISBN - 13:

978 - 0133943030