

Contents

1	Introduction	3
1.1	Terminology	3
1.2	Overview	3
1.3	Primary Features	4
2	Installation	5
2.1	Source Code Repo	5
2.1.1	Installing and running from source	5
2.1.2	Requirements	5
3	Getting Started	6
3.1	Single Search Mode	6
3.2	Submission Search Mode	9
4	Module information	11
4.1	Detection	11
4.2	Distinguishing	12
4.3	Deciding	14
5	Architecture and Source Files	15
5.1	System Architecture	15
5.2	Source Files	16

1 Introduction

1.1 Terminology

- Bot - A piece of software created to automate tasks. Commonly used for moderation and quality of life features (such as automatically spellchecking a users posts, or correcting grammar).
- Beneficial Bot - A bot dedicated to making the user experience better, a positive bot. Usually made by administrators or helpful community members.
- Malicious Bot - A bot dedicated to worsening the user experience, a negative bot. Usually made by third-party bad actors.
- Reddit - A social media site dedicated to users connecting with each other on various different topics.
- Subreddit - A section of Reddit dedicated to a specific topic (for example, there may exist a subreddit for programming, where users only discuss programming, or one dedicated to cooking, video games, etc.).
- Karma - Users on Reddit can 'upvote' or 'downvote' their fellow users posts, resulting in a loss or gain of karma. Karma is basically a unit of measurement used to see what the community at large thinks about a users post, with negative karma (downvote) being negative, and positive karma (upvote) being positive.
- API - Application Programming Interface (A set of rules and protocol that allows different software applications to communicate to each other.)
- PRAW - "Python Reddit API Wrapper", the python library that allows us to communicate with Reddit to receive our data for analysis.
- Spacy - Python library dedicated to text analysis.

1.2 Overview

This project is Enhancing Social Media Governance with Policing Bots (known as 'Policing Bots' from here on out. This project aims to help combat the ever growing bot problem on social media websites. The social media site of choice for this project is <https://www.reddit.com>. The project was made to help Reddit users and moderators / administrators detect bots and decide what to do with them. The project also aims

to detect the amount of maliciousness in a bot (defined by us as a bot that harasses users, tries to steal data from them, or are just generally unpleasant to other users). We wanted to be careful in this regard, as many bots are useful, and actually deployed by Reddit themselves to help moderate the site, or make the quality of life better for the user-base.

1.3 Primary Features

The key features of Policing Bots are broken into three core modules, which include:

- Detection Module - The primary module for detecting the existence of a bot. Used to tell if a user is a bot or a real human
- Distinguish Module - Module used to determine whether a bot is a beneficial bot or a malicious bot
- Decide Module - Used for advice on what should be done about the bot (whether to ignore, or take care of)

Each of these modules are described in further detail in chapter 4.

2 Installation

2.1 Source Code Repo

The source code for Policing Bots can be found at the following GitHub link:

```
1 https://github.com/Balkirprpl/PBot
```

The main branch is currently the live deployment.

2.1.1 Installing and running from source

Policing bots can be downloaded by using the following command in a terminal:

```
1 git clone git@github.com:Balkirprpl/PBot.git
```

2.1.2 Requirements

Policing Bots requires Python3 with the 'PRAW' and 'Spacy' libraries installed. These libraries can be installed by using the following commands in a terminal:

```
1 pip3 install PRAW
2 pip3 install Spacy
```

Policing Bots also requires a 'keys.py' file. The file should be structured as such:

```
1 key = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
2 client = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
3 user_agent = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

where the "x" are replaced with your developer information from Reddit. This information can be obtained like this:

```
1 Go to https://old.reddit.com/prefs/apps/
2 Scroll down and click on "create application", selecting "script".
3
4 Once the application is created, you must register it by following this
  link: https://old.reddit.com/wiki/api
5
6 When completed, a client ID and secret will be revealed, and you will
  come away with three different values:
7   User Agent = name of the app
8   Client ID = the string that is displayed under the name, in the "
  authorized applications" tab of the account settings.
9   Secret ( key ) = viewing the details of your newly created
  application will show a field with the title, "secret".
10
11 Store these values in a .py file called 'keys.py'
```

3 Getting Started

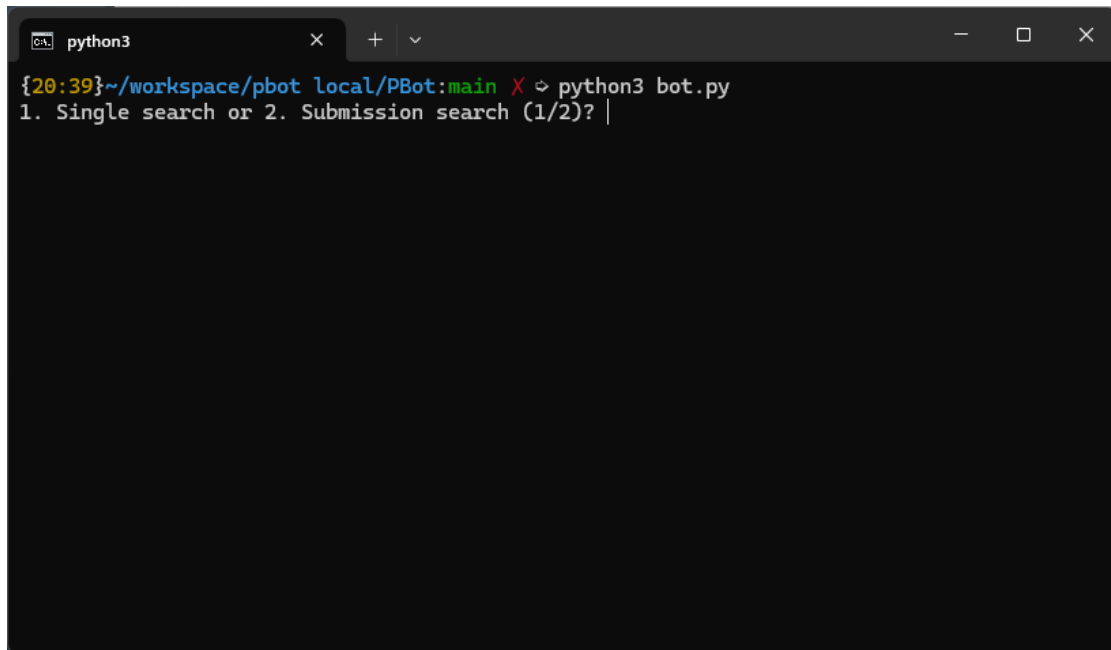
Policing Bots is started by running bot.py

```
1 python3 bot.py
```

To run the program against a list (for example, a list of suspected bots), the list MUST begin with a '1' on a newline and end with a '0' on a newline. Running the program against a list is as easy as doing this:

```
1 python3 bot.py < bots.txt
```

From here, the user can type '1' for the 'single search' mode, or '2' for the submission search. Details on the differences between the two to follow.



```
python3
{20:39}~/workspace/pbot local/PBot:main X ↵ python3 bot.py
1. Single search or 2. Submission search (1/2)? |
```

3.1 Single Search Mode

The 'Single Search' mode is used to search a single user by their username. When the selection is made, the user is prompted to enter a username to search, or to press '0' to cancel and go back to the single user or submission choice. Below is an example of the results of a single search mode search. The user being shown as an example is Bill

Gates' Reddit account. This is a known human account.

```
python3
{23:33}~/workspace/pbot local/PBot:main X > python3 bot.py
1. Single search or 2. Submission search (1/2)? 1
Username or 0 to leave: thisisbillgates
Username: thisisbillgates
Id: 91u3j
Link Karma: 164566
Comment Karma: 945447
Total Karma: 1142478
Account age: 18/09/12
Is verified: True
Total submissions: 121
Total comments: 460

1st Bot Detection Score: 0.2890025510320768 (Not Likely Bot)
2nd Bot Detection Score: 56.04396948339161 (Not Likely Bot)
Agreement in Detection

thisisbillgates is not believed to be a bot
>>> |
```

As can be seen, there are many metrics that are checked. A listing of what the fields mean are as such:

- Username: The users selected username
- Id: The users Identification number. Reddit assigns these to all users when they make their account.
- Link Karma: How much karma the user has gained based on posts they have created.
- Comment Karma: How much karma the user has gained based on the comments they have written.
- Account Age: The date the user signed up for Reddit.
- Is Verified: States whether the user has decided to confirm an email address with Reddit or not.
- Total submissions: The number of posts the user has created.

- Total Comments: The number of comments the user has written.
- 1st Bot Detection Score: This is a numerical value of our first detection algorithm results, more detail on that can be found in chapter 4 section 1
- 2nd Bot Detection Score: This This is a numerical value of our second detection algorithm results, more detail on that can be found in chapter 4 section 1

At the end, a message is printed to state whether or not both detection algorithms are in agreement on whether the user is a bot or not. In the above example, both algorithms believe the user is a bot. The next message "User is not believed to be a bot" only comes on if both of the algorithms are in agreement. After this is printed out, the user is able to search up another user, or press 0 to exit the program and get more information on what was found in this session.

The above image was done on an account that is a known human, below is an example of the single search running against a known bot (this bot is a bot used for moderation purposes, and is a known beneficial bot).

```

python3
{23:29}~/workspace/pbot local/PBot:main X ↵ python3 bot.py
1. Single search or 2. Submission search (1/2)? 1
Username or 0 to leave: AutoModerator
AutoModerator has been found in the list.
Username: AutoModerator
Id: 61423
Link Karma: 1000
Comment Karma: 1000
Total Karma: 2424684
Account age: 05/01/12
Is verified: True
Total submissions: 926
Total comments: 947

1st Bot Detection Score: 1.0 (Likely Bot)
2nd Bot Detection Score: 975.648924196836 (Likely Bot)
Agreement in Detection

Initiating further analysis
checking links
AutoModerator has 0 links and 0 shortened links
This account has used profanity 622 times
is this account a autodeclared bot? True

['AutoModerator:61423 was caught possibly harrassing another user']
>>> |

```

As can be seen, there are quite a few new pieces of information. This new information will only show up when one or both detection algorithms are flagging the user as a bot. In this case, both of the algorithms are in agreement for this user, which is good; because

the user is a known bot.

The new information given is pulled from the distinguish algorithm, and is as follows:

- **Checking Links:** The algorithm is looking for comments where the suspicious user sends links. There is very little reason why a bot should be trying to send a user to an external site. Shortened links are treated even more harshly, as the victim has very little clue for where they are being sent. This section counts the number of links and shortened links.
- **Profanity counter:** The algorithm is looking for excessive amounts of profanity / slurs being used. Some bots are used to harass users by being a disturbance with this kind of language. The algorithm searches through a word-list of this kind of language and counts it up. If the profanity reaches a certain threshold, we flag it as potentially spamming.
- **Does this bot declare itself as a bot?:** On Reddit, most of the bots that are created for beneficial purposes include a message at the end of their comments stating that they are a bot, and can opt out of seeing its comments. No malicious bot would tell you that they are a bot, so we take the existence of this message as a positive sign that the bot is not malicious.

A list of known bots is included in the repository, and if the bot is in the list, we tell the user that the user is a known friendly bot, and skip them.

At the end of the analysis, the framework alerts the user to whether or not it thinks the given user is a bot, and whether it thinks it is a malicious or beneficial bot. From here, advice on what to do with the bot is given.

3.2 Submission Search Mode

Selecting '2' on the beginning screen takes the user to 'Submission Search Mode'. The program will then prompt the user to type a subreddit name. Doing this will then prompt the user for New/Hot/Top submissions, where the user can select their choice by typing N for new, H for hot, or T for top. The differences between the three are as follows:

- **New:** Checks the newest posts posted to the selected subreddit
- **Hot:** Checks the posts on the posted subreddit that are seeing the most user activity at the moment
- **Top:** Checks the posts on the posted subreddit that have seen the most amount of user activity from all time

From there, the program will ask the user for the amount of posts to retrieve, this is self-explanatory. From there the last prompt asks the user for how much depth the program should go for. The amount of depth chosen is how many replies down in a comment thread it should search (2 would result in the program searching the comment, then 2 replies to each comment). Worth noting is that the amount chosen for posts and depth has an effect on the amount of time taken for the program to run. High numbers will result in a very long amount of runtime, as the program has to parse through all the data and pull it back. After that, it will go through all of the posts it finds given the restraints given and run the tool on the users it finds (giving the same data from the single search, just with more users). An example is shown below, using the 'gaming' subreddit, 'N', '3' posts, with a depth of '1':

```
python3
{0:01}~/workspace/pbot local/PBot:main X python3 bot.py
1. Single search or 2. Submission search (1/2)? 2
What subreddit you want to look at? gaming
New, Hot or top submissions? (N/H/T) N
How many posts to retrieve? 3
Depth: 1
Submission: https://i.redd.it/4bi7ccer4duc1.jpeg
Submission: https://youtu.be/N-EQsN63PBE?t=70
Username: Persies
Id: e2z3V7
Link Karma: 26174
Comment Karma: 148192
Total Karma: 176187
Account age: 31/07/16
Is verified: True
Total submissions: 309
Total comments: 1800

1st Bot Detection Score: 0.274483924884732 (Not Likely Bot)
2nd Bot Detection Score: 76.63708894649168 (Not Likely Bot)
Agreement in Detection
Persies is not believed to be a bot
Username: keeperofkey
Id: jnvcklv5
Link Karma: 971
Comment Karma: 2286
Total Karma: 3177
Account age: 28/06/18
Is verified: True
Total submissions: 6
Total comments: 593

1st Bot Detection Score: 1.629934194883242e-05 (Not Likely Bot)
2nd Bot Detection Score: 86.78372389138333 (Not Likely Bot)
Agreement in Detection
keeperofkey is not believed to be a bot
Username: Havi_jarnsida
Id: q0joya6v
Link Karma: 5
Comment Karma: 2004
Total Karma: 2019
Account age: 14/07/22
Is verified: True
Total submissions: 5
Total comments: 579

1st Bot Detection Score: 0.012853197543849526 (Not Likely Bot)
2nd Bot Detection Score: 20.0299835837616 (Not Likely Bot)
Agreement in Detection
Havi_jarnsida is not believed to be a bot
Username: HawthorneTR
```

4 Module information

Policing bots is run based on three modules, with information about each to follow.

4.1 Detection

This is arguably the most important module, as it is responsible for both of the algorithms we use to determine whether a user is a bot or a human. These modules are held in the files:

```
1 detect1.py
2 detect2.py
```

An important note is that when working on bot vs. human detection, there is no perfect algorithm. The best thing to do in this situation is to create multiple detection algorithms for bots and to evaluate them manually once analysis is done. At the current level of computing, some human review is always going to be needed.

Knowing this, the first detection algorithm is based on text frequency analysis. We use the 'similarity' function in the Spacy library. We have a function scan_comments:

```
1 def scan_comments(comment_similarity_array):
2     # -----
3     #   Calculating txt similarity
4     # -----
5     z = 1.0
6     l = len(comment_similarity_array)
7     for i1 in range(l):
8         for i2 in range(i1 + 1, l):
9             c1 = comment_similarity_array[i1]
10            c2 = comment_similarity_array[i2]
11            z *= compare_text(c1, c2)
12
13     return z
```

This scans the previous 5 comments the suspected account has posted and checks how similar the text was. This is given a z-score and is analyzed based on how close the z-score is to 1.0, with a score of 0.3 or more getting flagged as a bot. An important note is that 5 comments is just what we chose for convenience, the more posts chosen will increase accuracy, while heavily increasing the amount of time it takes for the program to run.

The second detection algorithm is based on a variety of user information that the Reddit API gives us with the PRAW library. The second algorithm searches through the following metrics:

- Post Interval: How frequently a user posts, no human should be posting new comments / posts within milliseconds of each other.
- Karma: How much positive or negative karma a user has, this one has a small weight, as it could just be a disruptive human. It IS checked though, as many users will downvote a bot if they notice it
- Account Age: An account that has made posts and has been around for a long time is less likely to be a bot, as most bots get banned rather quickly. It is very rare that a bot (that actually interacts with the site) survives longer than a month; so we check for that.
- Verified Email: Reddit gives the option to verify an email with your account. A bot will usually not have a verified email, so this is a cause for suspicion.
- In our lists: If a bot is already in the list of bots we have, this is automatically tripped and we don't need to analyze any further.

Each of these categories has a weight associated with it with known bot being the highest (automatic flag) and Verified Email / Karma being the lowest weight. All of the metrics are given a number and a weight, these are then added up and checked against a 'bot score'. If this number exceeds 130, we flag it as a bot. This number could be tweaked if we feel it is too sensitive, or not sensitive enough. Below is the function `bot_score()` that calculates this:

```
1 def BotScore(username, PostLimit):
2     user= reddit.redditor(username)
3     totalscore = 0
4
5     totalscore += AnalyseAccount(user)
6     totalscore += AnalysePosts(user,PostLimit)
7     totalscore += PostingInterval(user, PostLimit)
8     totalscore += AnalyseComments(user, PostLimit)
9     totalscore += CommentInterval(user, PostLimit)
10
11     return totalscore
```

4.2 Distinguishing

The 'distinguish' module is made for accounts that we have analyzed to be bots. From this module, the program runs a series of simple checks to determine whether the bot should be flagged as a beneficial bot or a malicious bot. This is done through three main checks. The three checks are as follows:

- Links and shortened links: There is very little reason why a bot should be sending a user off of reddit.com. There is no way to verify if the link they are sending the user to is a real website or a scam, so we look at links extremely poorly. This is heavily weighted if the links have been run through a link shortener. A link shortener will hide the URL of the link, so the user is truly blind. The existence of a shortened link is almost instantly a red flag for maliciousness. We check for popular link shortener websites using this list in python, a comment containing these URLs is flagged.

```

1 SHORT_LINKS = [
2     'bit.ly', 'tinyurl.com', 'goo.gl', 't.co', 'ow.ly', '
  buff.ly',
3     'adf.ly', 'is.gd', 'cutt.ly', 'shorte.st'
4     ]
5

```

- Profanity / Slurs: We have supplied a word-list of vulgar language. When we check the bots comments, we take note of how many of our words in the list show up in their recent comments. If it hits a certain threshold, we warn that the account may be spamming people. There is very little reason to use excessive profanity, and no reason to be using slurs. This can be seen in the function 'count_bad_words()' shown below:

```

1 def count_bad_words(bot):
2     comments = bot.user.comments.new(limit=None)
3     bad_word_count = 0
4     for comment in comments:
5         # split comment into all comments
6         comment_words = comment.body.lower().replace('|', ' ').
  split()
7         # sum all the occurrences of a bad word in the comment
8         for word in comment_words:
9             if word in BAD_WORDS_SET:
10                bad_word_count += 1
11
12     return bad_word_count
13

```

- In list: We already know that the bots in the list are beneficial, so they will not be flagged as malicious (although we still warn the user about flags that have been tripped about the top two checks).

More work is to be done on the maliciousness front. The next step we brainstormed is a function that checks if a suspected bot has posted the exact same message across a variety of sub-reddits. If a user posts a message in the programming sub-reddit and the cooking sub-reddit, there is very little chance that the message content is relevant to both of those subjects. Too many of these would result in the algorithm flagging and letting the user know.

4.3 Deciding

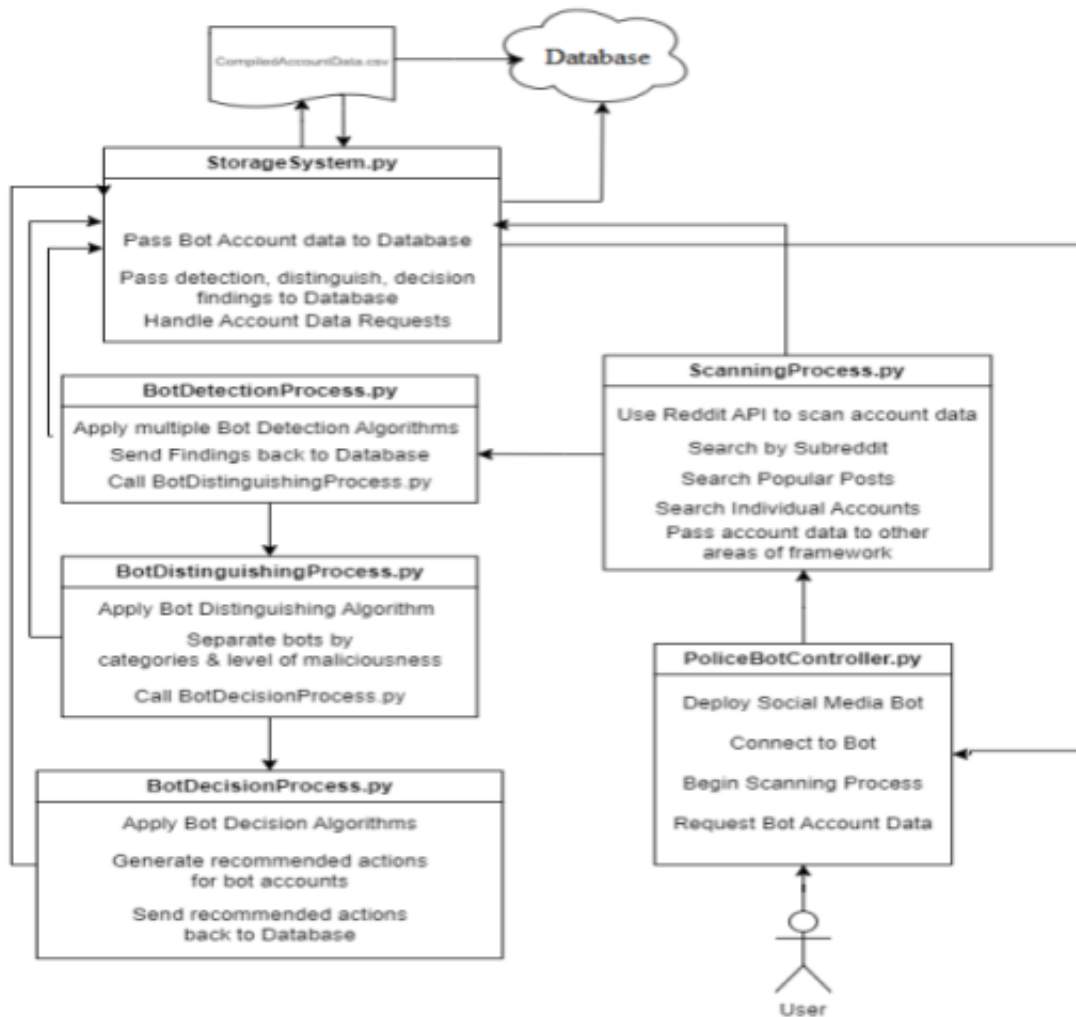
The final module is for the user to decide what to do with a bot that has been detected. This part must be done manually, as bot detection is not a perfect science and we don't want to report users who the program flagged incorrectly. This module is very simple, it provides the information found about the user, a link to their account, and instructions on how to report them to the Reddit Administrators / Moderation team. The code for this can be found below.

```
1 link = "https://www.reddit.com/user/"
2 cyan = '\033[34m'
3 reset = '\033[37m'
4
5 def decide(account, ignore_list):
6     if 'exiting' in ignore_list:
7         print(f"url: {cyan}{link}{account.name}{reset}")
8         return True
9     if 'all' in ignore_list or account.good_bot:
10        return False
11    if account.good_bot == '0' and not 'good' in ignore_list:
12        print(f"Autodeclared bot usually not harmful.")
13        x = input("Want to report? (y?) (0 to ignore all)")
14        if x.rstrip().lower() == 'y':
15            print(f"url: {cyan}{link}{account.name}{reset}")
16            return True
17    else:
18        if len(account.reasons) > 0:
19            print(f"url: {cyan}{link}{account.name}{reset}")
20            return True
21        elif not 'inconclusive' in ignore_list:
22            print(f"No decisive reason to report this bot.")
23            z = input("Want to report? (y?)")
24            if z.rstrip().lower() == 'y':
25                print(f"url: {cyan}{link}{account.name}{reset}")
26    return False
27
```

The end result of this allows the user to continue searching, or to close the program and print out all of the results that were found.

5 Architecture and Source Files

5.1 System Architecture



Shown above is a rough system architecture diagram that showcases how the project operates as it runs through all of the modules and sections of the program. Some of the names are a little different in the final project, but the control flow remains the same.

5.2 Source Files

- bad-words.txt: This is a word-list of known profanity/slurs. This list is referenced in the distinguish algorithm when searching for spamming bots. This list was pulled from:

```
1 https://www.cs.cmu.edu/~biglou/resources/bad-words.txt
```

- bot.py: The primary resource of the project. Runs the decide, detect, and distinguish models and is the entry point to the project.
- bot_db.sql: The SQL schema for the database that holds the results we find from searching for users or submissions.
- bots.txt: A word-list of known beneficial bots on Reddit, used for testing for false-negatives and for ignoring if they come up in a search (don't need to test for maliciousness on them)
- database.csv: A csv file of all of the user-data we are pulling from Reddit. All of the results we find are stored in this file.
- database.py: This python program is used to populate the database (database.csv) with the given schema (bot_db.sql). This is how the data is actually sent to the database.
- data_request.py: Does the opposite of database.py, dumps the database into a csv file. Used primarily to analyze data and allow users to parse data without use of a database module
- detect1.py: The python file used for the first detection algorithm
- detect2.py: The python file used for the second detection algorithm
- distinguish.py: The python file used for the distinguishing algorithm
- decide.py: The python file used for the decide module
- requirements.txt: A text file used to install the required libraries needed for bot.py
- colors.py: A list of ANSI escape sequences for coloring important text in the program.