Bot || !

Patrick Canny, Lane Gramling, Liam Ormiston, Damian Vu, Taylor Walenczyk

Project Synopsis

“Bot || !” leverages existing and user-generated data to classify pathogenic social media accounts. Its applications include identification and removal of spam accounts that influence public opinion.

Project Description

In recent years, pathogenic accounts have been used as a tool for influencing the perception of the public. For instance, such accounts significantly impact the news people see every day, skew perceptions on important global events, sow discord into populations, and ultimately undermine the effectiveness of the democratic processes. The ebb and flow of the battle against pathogenic social media has blossomed into a vibrant field. "Bot || !" seeks to aid in the fight against the far-reaching negative impacts of pathogenic social media by employing an innovative approach to the classification of pathogenic accounts.

Humans have the uncanny ability to solve incredibly complex problems quickly. Uber tapped into this innate ability to classify signs for their self-driving car technology. Likewise, we will we use the power of human perception to classify pathogenic social media accounts, effectively crowdsourcing the training of a pathogenic account classifier. It is possible this novel approach will lead to a substantial contribution to the field. Once our classification system is sufficiently accurate, tech companies and intelligence agencies alike can utilize the data "Bot || !" produces to further their investigations.

The complexity of the project affords us the opportunity to develop our skills across the tech stack. We will integrate a sublime user-experience, robust data storage systems, and a novel analytics engine to support the training of our classifier. By March, we will have a mobile and web application where users can the authenticity of an account. Their decision will work its way into our analytics engine to produce an effective data stream for our classifier. This process will iteratively tune the classifier until it can confidently judge the authenticity of a social media account. The data we select for analysis will combine existing techniques with algorithms developed from our understanding of the psychology behind how people interact with social networks and respond to content. This approach will allow us to extend the capabilities of our algorithms and add features as far as our timeline allows.

Project Milestones

Milestones are integral to the success of the *Bot || !*. After thorough debate, we decided on a series of events that serve as effective milestones. The first is the creation of a comprehensive project timeline. We will gain invaluable insight into the demands of the project while building it; From then on, it will serve as a beacon throughout the year. By the onset of November, the timeline will be complete. Additionally, we will generate supporting UML documents, i.e., preliminary wireframes, for all project components. By the close of the first semester, will have a minimally viable version of our product. This entails simple mobile and web applications; simple user registration; a simple API for minimal interaction with the database; and starter datasets for non, potential, and confirmed pathogenic accounts.

When we move into the implementation phase, the milestones will be broken down on a monthly basis. In February, the application should be able to register and store users in a database, along with any necessary information about those users. We will also have a functioning build system set up, in order to ensure that our product is tested and functional with each change. By the end of March, our application frontend will be more refined and fully featured. Our Internal API will be ready to transfer data across each piece of the application to the backend toolkit, which will process that data and continually classify accounts. Finally, by the end of April, we will have deployed an application that is robust and able to effectively classify newly created accounts as pathogenic or not.

Project Budget

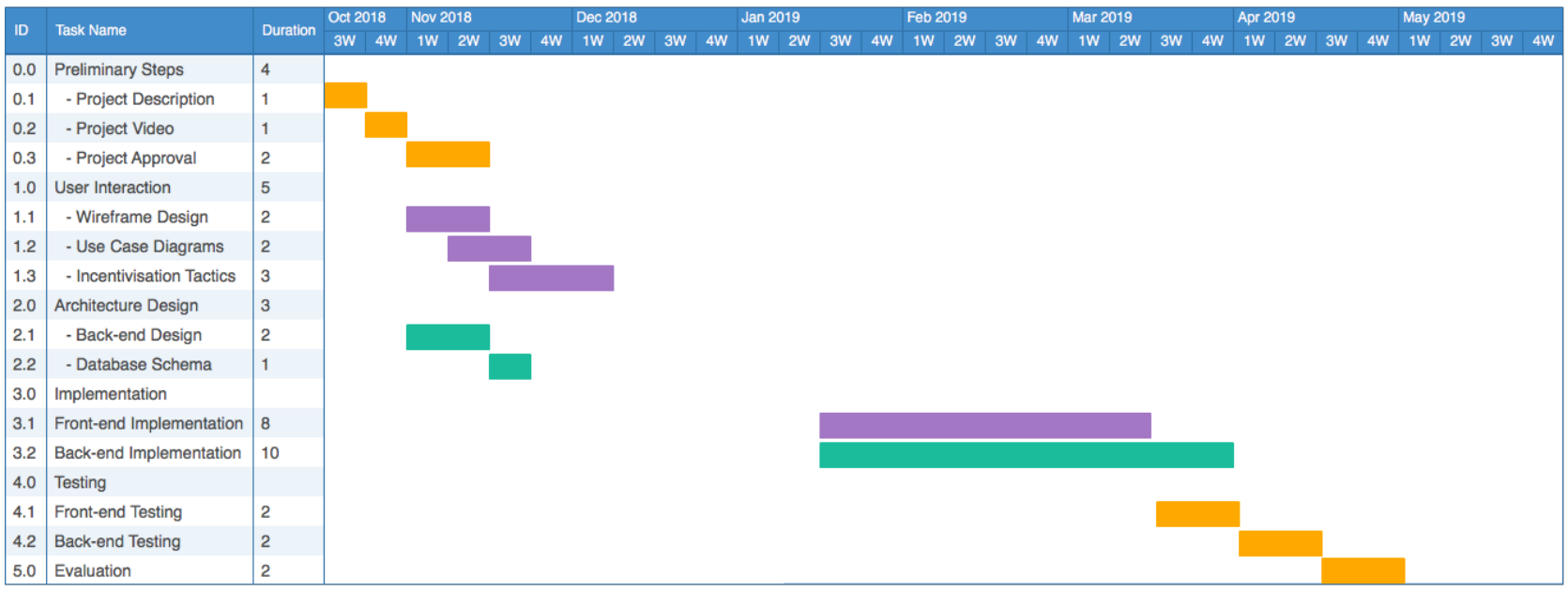
Item	Price	Quantity	Timeline	Description	Purpose
Dell Optiplex 790 Refurbished high-performance MiniTower	$237.89	5	Needed near the beginning of development to aggregate data	Important specs: 8GB RAM, 2.220GB storage, 3.4Ghz Intel process Full description	Hardware for a small HDFS cluster for data storage and processing
Sketch	$99	1	Upon Approval	Digital Design Software	Enables us to create a template for our UI
Google Play Store	$25	1	Near completion	Publishing permission to	Publishing

				the Google Play Store	
Apple Store	$99	1	Near completion	Publishing permission to the Apple Play	Publishing
Total	$1412.45				

Work Plan/ Roles

Liam - Application developer, front-end engineering
Damian - Deployment/build engineer, data visualization, internal API developer
Lane - Backend engineering, internal API developer, data engineering
Pat - Data engineering, data modeling, application development
Taylor - Backend engineering, data modeling, Hadoop developer

Gantt Charts



Preliminary Project Design

"Bot || !" has five main components. They are the frontend, API, data storage, analytics, and data aggregation. Each piece functions independently yet contributes to the overall mission in significant ways. In a nutshell, data comes into the system from users via the frontend and from social media services on the backend. The flow of accounts from front to back via the API continually refines our classifier. The classifier then produces conclusive data regarding our data lake on a frequent basis for the use of interested parties.

More precisely, here is a walk-through of the life of datum in our architecture. We will store pre-existing data on an HDFS cluster that we will use for our mobile application. The data that we will store on the cluster will be a culmination of twitter accounts that have been identified as pathogenic, non-pathogenic, and unidentified through. This will be done through Twitter API as well as a Data Collection Service as shown in **Figure 1**.

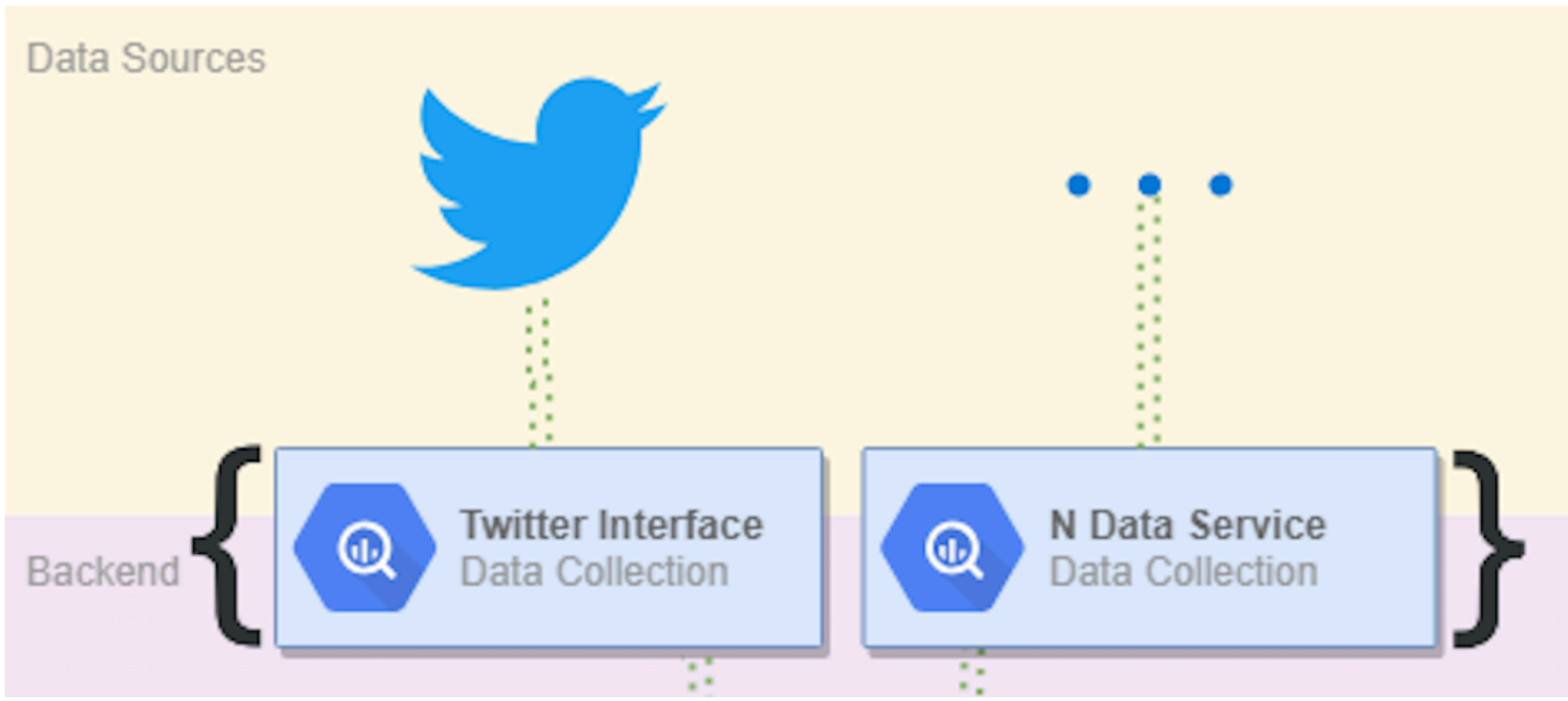


Figure 1

The application will take that data and present an account to a user as shown in **Figure 2**.

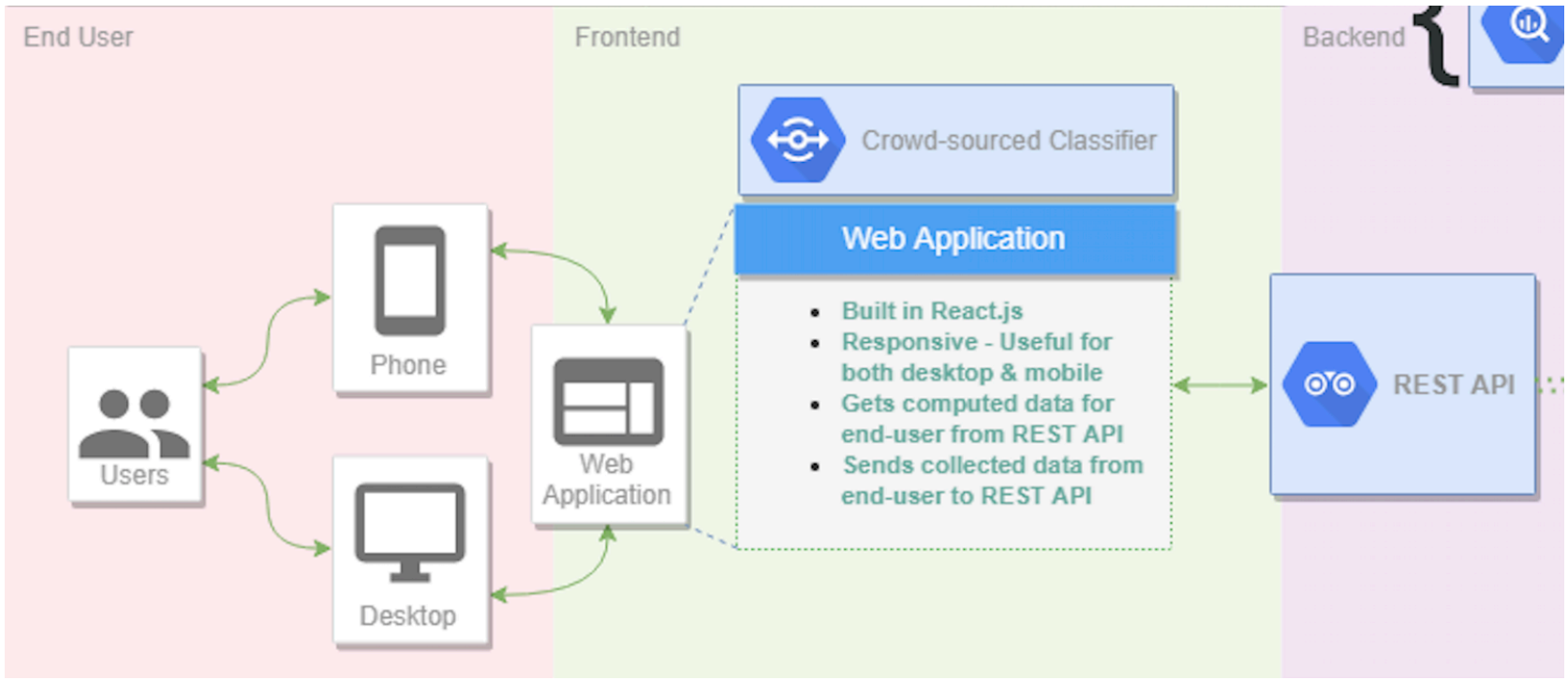


Figure 2

The user will then have the option to identify the account as either pathogenic or non-pathogenic. The user's response is then sent back to the cluster and stored in a separate dataset with the identified field. This process can be seen in **Figure 3**.

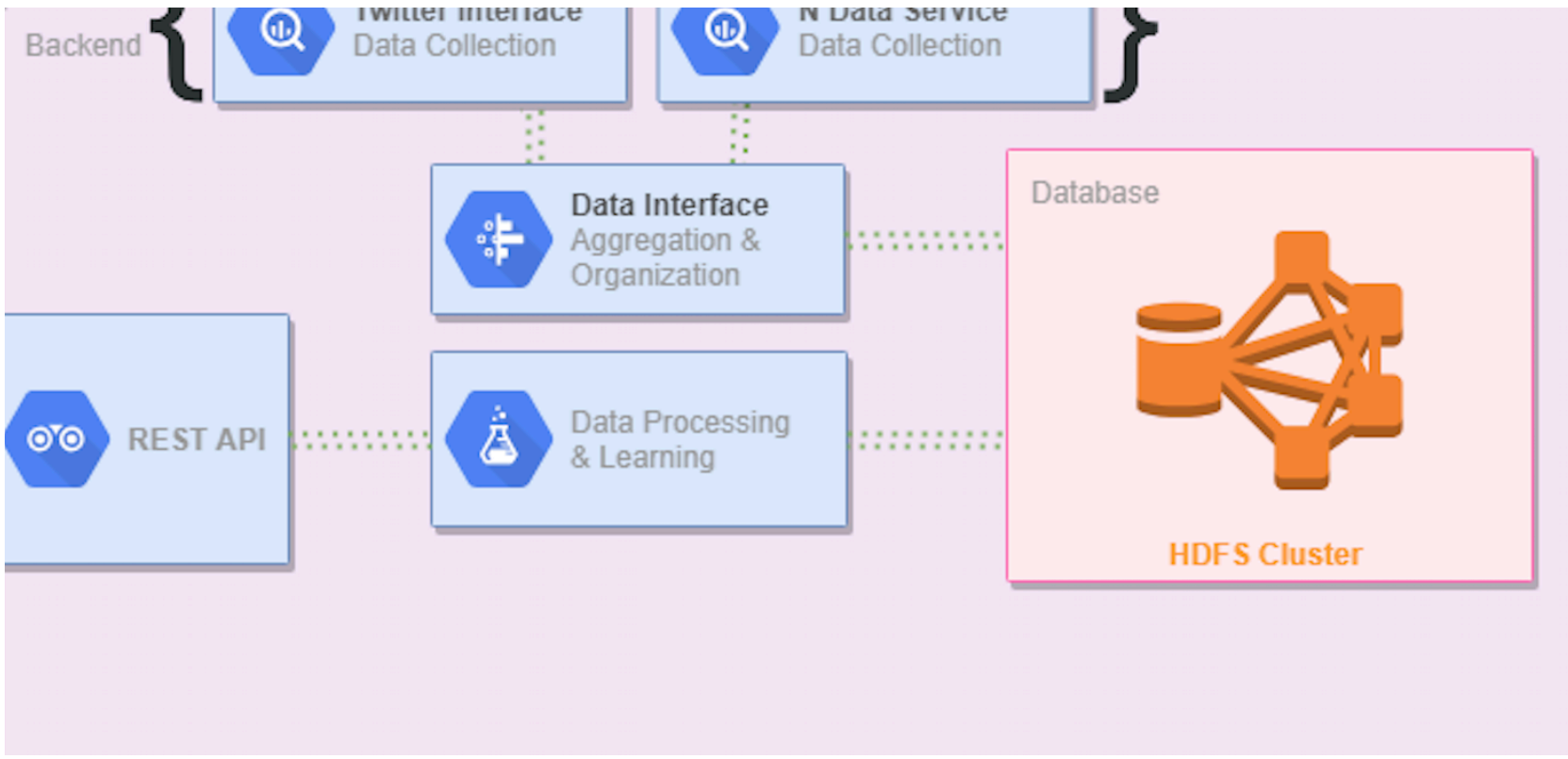


Figure 3

From there, we will analyze this modification to the dataset to generate the classifier.

The frontend application will be built on top of a web-based language that can be ported to mobile environments. ReactJS and React Native will be the technologies underpinning the frontend. ReactJS is a Facebook language designed to quickly and easily create components that "react" with different environments the application may exist within. It perfectly fits our use case by allowing us to create a single application for web, Android, and IOS. React Native then

takes the React code and converts it into native code for each mobile platform. The application will be made up of four different views. The first view the user will see is a account creation / login screen. The user will be able to log in or create a new account if they do not already have an account. The second view will be the main view. The user will see a username, profile picture, bio, and the last five tweets that an account has tweeted. The user will then be able to swipe the view right to classify the account as pathogenic or left to classify the account as non-pathogenic. When the user swipes to make their decision, they are then presented with another account's information that will continue the classification process. The newly classified account is then sent to the cluster to be analyzed and categorized. The third view will be the leaderboard of how the user stacks up to other users on the app. For each correct classification, where an account has already been identified, a user will receive +1 point. If they incorrectly classify an account, they will receive -1 point. For every classification that was unidentified previously but two users come to the same identification, both users receive +5 points. The final view will be an account view where the user can see and edit their account information.

If our application is to have any notable impact, we will need to develop an effective framework for aggregating data from a variety of sources. As displayed in Figure 1, we will primarily mine account data from Twitter. To do so, we will concurrently utilize a set of API keys to maximize our data-mining efficiency. We will use the Python learning library (described below) to guarantee that the accounts the usefulness of the accounts we mine. This aggregation layer is also where we will investigate the merit of bringing in data from other sources, and the entry point for all data to be consumed by our system. This will involve setting up some sort of collector that is listening for data at the sources we specify and then adding that data to the cluster. This is advantageous since it is easily extensible to any set of sources that we may find valuable in the course of our work.

One of the crucial pieces of our application will be the data processing and analytics layer. In this piece of the application, we seek to use machine learning algorithms alongside data science models of our design to derive new meaning from the data we will have at our disposal. The primary means of evaluating and manipulating this large dataset will be a custom Python library that will be created solely to run machine learning algorithms on the training data and to serve the results reasonably. When compared to other spam account classifiers, one distinguishing characteristic of our software is the integration of data collected through user interactions. Though innovative, we must account increased uncertainty in our system. We clean and process our incoming data to ensure the succinctness of our datasets, and maximize their utility. This process will be a feature of the library, along with a set of machine learning models that the data can be passed through for training and testing. Ideally, we will work towards a position where we can update our models each time a new piece of data comes in, resulting in the most current real-time processing. Of course, with Python, speed will be a concern in this layer in particular. In addition to this library, we will need to find a good way to visualize and interact with our models. This is where using a tool such as Jupyter Notebook or Tableau comes into play. Finally, this inference layer of our application will need to be able to communicate with both the backend data storage layers, as well as the REST API. We will store

our modified and/or generated datasets in our database and HDFS cluster in order for our previously processed data to take effect in future iterations. We will also need to allow for access between the REST API and our data library in order to intelligently serve up data to the frontend, and not pass data that will not be useful to be manipulated. As with most portions of our application, we will need to host this service somewhere. Ideally, this will be through a paid AWS or Google Cloud account, but if we cannot obtain access to these accounts we will be able to host for up to a year on either or both of the platforms previously mentioned.

The final essential aspect of our application is the data storage layer. We will house data in two components: an HDFS cluster and some relational database management system. The former is integral to our data analytics platform; the latter will interface with our API. The Apache Foundation provides excellent free software to spin-up and to manage a healthy big data cluster. It is horizontally scalable and is optimized to run batch jobs across large, unstructured data sets. We will use Postgresql for our RDBMS. It is a popular, well-documented language and well suited for our needs. To move data between the two, we will use Sqoop, another Apache Foundation technology. Sqoop is designed to efficiently move data between sources and perfectly fits our use case.

Design Constraints

Programming Language:

For our project, we aim to make our app accessible to everyone. Because of this, we need a language that can work on all platforms - web, iOS, and Android. Most companies would have different departments in charge of each platform using native languages for each. However, we are only a small team that cannot dedicate such time and resources to a developer with native languages for each platform. Therefore, we chose to use a web-based language that can adapt to native environments for iOS and Android such as React. Although using a language platform that can be used to build native mobile apps is useful, it will never be the same as developing with the native language. Understanding this constraint, we will strive to create an app that does not require a lot of native features. The fewer native features our app requires, the easier it will be to program in a non-native programming language. It will also give our app a more native experience.

Data Aggregation:

In order to build an effective classification algorithm, we need social media accounts. The multiplicity of social media platforms at our disposal solves this problem; In return, it presents a set of challenges. We must develop a data aggregation engine that takes input from multiple sources, i.e. Twitter, Facebook, Instagram, etc.; and condenses it into a consistent format and publishes the data to our HDFS cluster. Our schema must account take into

consideration the multitude of information at our disposal. Therefore, we must design it with the utmost care.

Real-time Needs:

Because “Bot || !” interfaces with real end users, latency is vital. A Hadoop cluster offers an excellent means to manage large amounts of data. However, it alone is not the solution. We need a relational database to interact with our API in real-time. The management of both data storage systems will be integral to the success of the projects. We must consider when and how to move data efficiently between the two.

Ethical and Intellectual Property Issues

Given the gravity of our subject, we must consider a myriad of ethical and intellectual property concerns. Fortunately, our project trods well-marked territory, so similar projects have been done previously. Though helpful, it presents a new set of challenges. We must be mindful of our progress and ensure that we are not infringing on the intellectual property of others in the pursuit of innovation.

Since our project partially deals with gathering user data, we will need to implement terms of use between ourselves and users. This is to promise to our users that we will not utilize their personal data outside of account lookup. On our end, this also means that a secure way to store user information without actually being able to access that information will need to be utilized. This could be down through some form of hashing, or through utilizing some pre-existing tool. Similar project designs to our proposed project exist, so a key ethical concern for our team will be ensuring that our project is unique and distinguishable from the currently existing projects. This will be primarily addressed through implementing a hybrid approach between traditional data collection and crowd-sourced data collection and uniquely leveraging both of these sources appropriately. Some of the key contributors we will need to be mindful of are the folks over at the University of Indiana. These people have been doing some development in the area of classifying pathogenic social media accounts, and a lot of their applications are in the same realm as ours. We will want to be good stewards of the profession and ensure that the work we will be doing will not be conflicting with their previous work. We will also seek opportunities to reach out to them if we end up in a situation where it seems that our two concepts seem to be more closely related. Our project also deals closely with how spam accounts impact public opinion, so it will be important that we are not serving up accounts that may be closely tied to hate speech or any derogatory content towards users. We will need to closely curate the accounts we push out to users in order to ensure that the accounts being served are ethically sound.

Preserving our personal IP while also seeking to make a contribution to the field is something that must be kept in mind as we move forward into creating our project. We would

like to be able to make a small mark on the field of data science in some way, either by contributing new datasets, algorithms, or a mixture of both. Since we seek to share our findings, we will likely be publishing our work to a public GitHub repository and sharing any information we find with a subset of the data science community. In our initial terms of use and agreement, we will need to add clauses designating how our software may be used and further distributed. A good place to start with this is licensing our project under a public license such as the MIT license. We will also identify potential places where we may be infringing on others' IP, and do what we can to ensure that we are using all forms of software and/or datasets ethically. Some potential places where these concerns need apply are any use of an ML platform, HDFS distribution information, use of public or private datasets and the like. We will also need to ensure that any non-original code that we choose to use is well documented and cited in our work.

Change Log

References

"Early Identification of Pathogenic Social Media Accounts" (Alvari, Shaabani, Shakarian)
"Uncovering Large Groups of Active Malicious Accounts in Online Social Networks" (Cao, Yang, Yu, Palow)
"Maximizing the Spread of Influence through a Social Network" (Kempe, Kleinberg, Tardos)
"A Randomised Controlled Trial of Social Network Targeting to Maximize Population Behaviour Change" (Kim, Hwong, Stafford, et. al.)
"Detecting Influence Campaigns in Social Networks Using the Ising Model" (des Mesnards, Zaman)
"The Rise of Social Bots" (Ferrara, Varol, Davis, Menczer, Flammini)