Mitigating Fake Digital Media and Quality Assurance

Hetesh Sehgal

Xinyu Chen

Kyle Jiang

SANTA CLARA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: June 9, 2020

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Hetesh Sehgal
Xinyu Chen
Kyle Jiang

ENTITLED

Mitigating Fake Digital Media and Quality Assurance

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

[illegible]

Thesis Advisor

N. Ling

Department Chair

Mitigating Fake Digital Media and Quality Assurance

by

Hetesh Sehgal
Xinyu Chen
Kyle Jiang

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 9, 2020

Mitigating Fake Digital Media and Quality Assurance

Hetesh Sehgal
Xinyu Chen
Kyle Jiang

Department of Computer Science and Engineering
Santa Clara University
June 9, 2020

ABSTRACT

With the introduction of social media, the internet is filled with an excess of data and content. Users feeds are cluttered with fake, malicious, and unnecessary information, polluting their page and wasting their time. As observed in the 2016 US election, spam accounts posting fake news were successfully able to sway political opinion and misinform the general population. Additionally, with social media becoming one of the biggest advertising markets, there is a rise in the number of fake accounts with a large number of followers that mainly consist of bots. It should be a priority to protect the public from false information and businesses that want to make money on social media platforms. Through utilizing Natural Language Processing, image recognition, and recommendation systems, powered through AI and Machine Learning, the goal of our project is to provide the user with content that is verified and tailored to their liking. This report details our plan to mitigate the amount of unnecessary content displayed in front of the user, and the rationale for our designs. It provides a guide for all the completed work, future iterations, performance results, and the reliability of our model as an efficient solution.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Our motivation is to prevent the spread of fake news and information, to ensure the public is well informed and gain control over what they choose to view.

1.2 Solution

With the introduction of social media, the internet is filled with an excess of data and content. Users feeds are cluttered with sometimes unnecessary information, polluting their page and wasting their time. Additionally, as observed in the 2016 US election, spam accounts posting fake news were successfully able to sway political opinion and misinform the general population. In just May and June of 2019, twitter suspended 70 million fake bot accounts[1]. Additionally, "researchers studied almost 250,000 social media active users who discussed the US elections both in 2016 and 2018 and detected over 30,000 bots"[2]. This surplus of false information has the potential to fuel national conflict, as well as manipulate individuals into buying into a false ideology.

With all the data and issues stated above, it is difficult for people to get trustworthy information from social media. People are exposed more to social media than traditional media. The false information they receive from it may subconsciously affect people's daily lives. Also, with social media becoming one of the biggest advertising markets, there's a rise in the number of fake accounts, with a large number of followers which mainly consists of bots. It is very important to protect the public from false information and businesses that want to make money on social media platforms. In this case, the actors and stakeholders consist of the general public, as well as individual entities looking to potentially advertise their product using a third party actor on social media. It is in the interest of the greater good of the public to ensure that the information they see online is genuine, and not fabricated by a malicious actor.

The existing solutions are built-in systems that social media platforms inherently have. As evidence, although they

[1] https://www.washingtonpost.com/technology/2018/07/06/twitter-is-sweeping-out-fake-accounts-like-never-before-putting-user-growth-risk/
[2] https://techxplore.com/news/2019-09-bots-harder-elections.html

are able to crack down on fake accounts after the fact, there is still an abundance of accounts that are uncaught. Using Artificial Intelligence and Machine learning, we plan to only recommend or show social media accounts that are of interest to the user and are authentic. We plan to do this using various metrics to gauge the authenticity of a social media account, as well as how closely it fits the preferences for the user. Our solution surpasses existing solutions because it only selectively shows profiles relative to the user's interests, whereas social media platforms such as Twitter or Facebook are less personalized as they focus more on providing an influx of data to keep the user engaged, rather than strictly displaying data tailored specifically to that user.

Chapter 2

Requirements

2.1 Functional Requirements

- The software will allow the users to check the authenticity of a news-based tweets from twitter.com.
- The software should be able to score the authenticity/trustworthiness of the news-based tweet, and display corresponding labels.
- The software should be able to display links of reference of other news articles that are similar to the news-based tweet.
- The software should be able to check the trustworthiness of the user who post the news-based tweet.

2.2 External Interface Requirements

2.2.1 User interface

- The software will be deployed as a web app.
- There should be a text box allowing the user to input the URL for the tweet, alongside a button for the user to submit the URL.
- Once the user clicks on the button, they should be directed to information relative to the content of the tweet.
- The information shown contains validity score of the news-based tweet that the users what to check, the trustworthy score of the author and link of similar news that is potentially related to the new-based tweet.

2.2.2 Software Interfaces

- The software is connected to an API that relays information to and from the server. The server contains our Machine Learning models that compares and scores the authenticity of the new-based tweet, and the trustworthiness score of its author. The API feeds the tweet URL to our back end server that contains processing code, and our models. The contents from the tweet URL are scraped, and fed into the models. The models then produce the tweet reliability score, account score, and related news links. This information is then fed back into the API, which feeds the information to the front-end where it is displayed on the web-app.

2.3 Other Nonfunctional Requirements

2.3.1 Performance Requirements

- Deployed as a web application
- Tweet URL used as query
- Accessible through desktop web-browsers

2.4 Constraints

- The software should be only accessible from Google Chrome web-browser with version 78 and above.

Chapter 3

Diagrams

3.1 Use Cases

Our interface comes in the form of a web app. The user simply copies the tweet URL of their tweet, and enters it into the text box in our web app. The user clicks the button on our web app and check the validity of a tweet. Once clicked, a tweet reliability score with the corresponding reliability label will appear alongside links to news articles that potentially back the content.

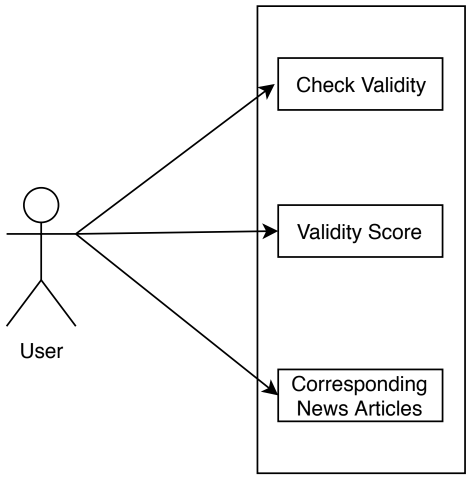


Figure 3.1: Use case diagram depicting interaction of user between interface

- Use Case 1: User wants to access web-app
  - Actor: User
  - Goal: Access web-app
  - Preconditions:

    * Web-app is deployed on a server or locally with an accessible IP/domain
  - Postconditions:
    * User accesses web-app through entering IP/domain in browser window.

- Use Case 2: User wants to check validity of tweet/content
  - Actor: User
  - Goal: Check validity of tweet
  - Preconditions:
    * User has access to web app.
    * User viewing twitter feed and copies URL
  - Postconditions:
    * User clicks on corresponding button and views accuracy score and label.

- Use Case 3: User wants to check news articles that potentially back up tweet/content.
  - Actor: User
  - Goal: Check corresponding news websites that back viewed tweet/content.
  - Preconditions:
    * User has access to web app.
    * User viewing twitter feed and copies URL
  - Postconditions:
    * User clicks on corresponding button and views links to relative news websites next to Tweet Reliability.

- Use Case 4: User wants to view author account rating.
  - Actor: User
  - Goal: Check corresponding news websites that back viewed tweet/content.
  - Preconditions:
    * User has access to web app.
    * User viewing twitter feed and copies URL
  - Postconditions:
    * User clicks on corresponding button and views author account score

3.2 Activity Diagram

The following diagrams represent the various paths/flow the user will take in order to accomplish a set goal.

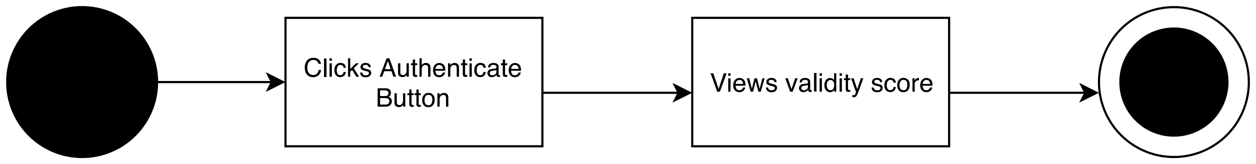


Figure 3.2: User attempts to view authenticity of tweet by clicking corresponding authenticate button

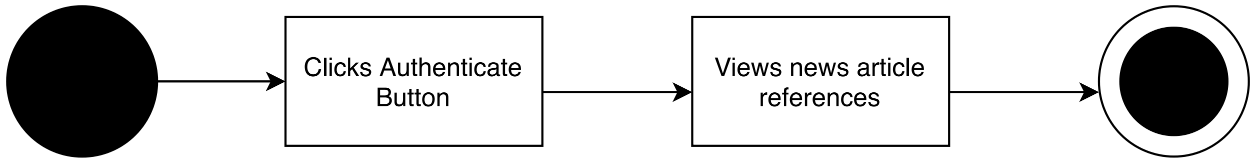


Figure 3.3: User attempts to view article reference links that back content of tweet

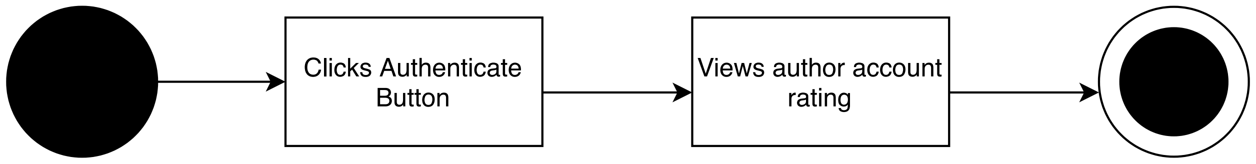
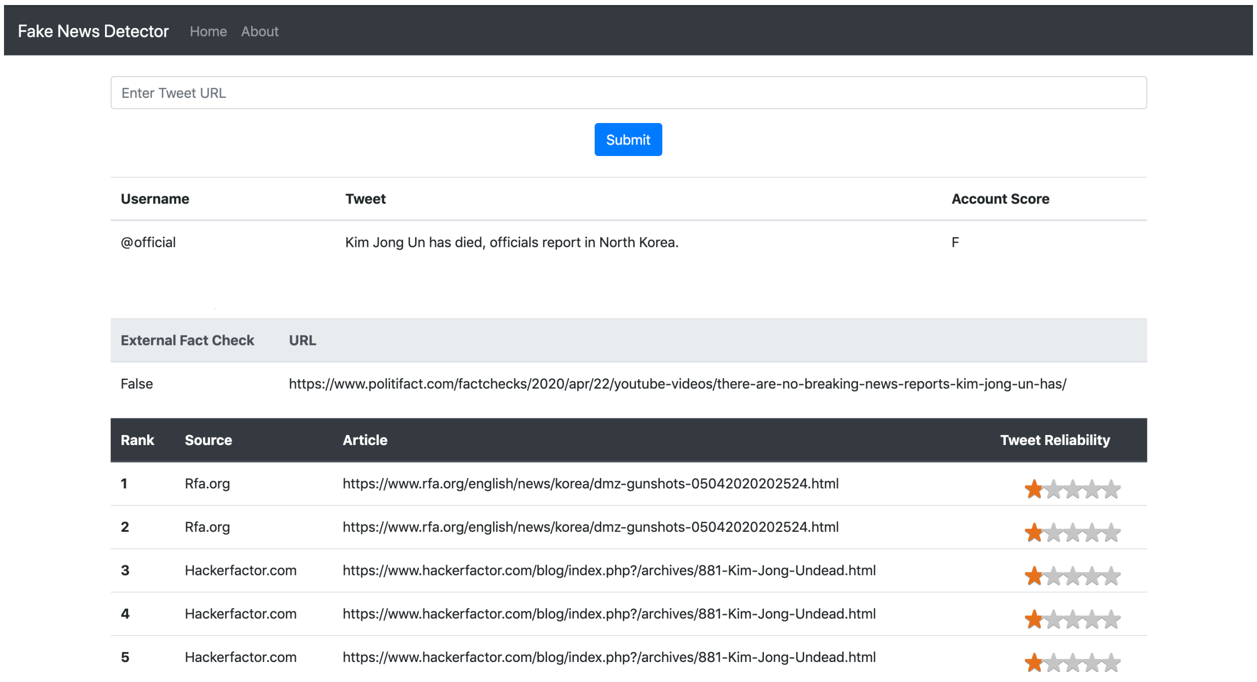


Figure 3.4: User attempts to view author of tweet/content account rating

3.3 Conceptual Model

Here we include a model of how out web-app currently looks like. There are two core User Interface components. One is the authenticate button (submit) that is placed underneath the text box to enter in the tweet URL, and the other is the information displayed when that button is clicked. Both features are illustrated below.



Username	Tweet	Account Score
@official	Kim Jong Un has died, officials report in North Korea.	F

External Fact Check	URL
False	https://www.politifact.com/factchecks/2020/apr/22/youtube-videos/there-are-no-breaking-news-reports-kim-jong-un-has/

Rank	Source	Article	Tweet Reliability
1	Rfa.org	https://www.rfa.org/english/news/korea/dmz-gunshots-05042020202524.html	★☆☆☆☆
2	Rfa.org	https://www.rfa.org/english/news/korea/dmz-gunshots-05042020202524.html	★☆☆☆☆
3	Hackerfactor.com	https://www.hackerfactor.com/blog/index.php?/archives/881-Kim-Jong-Undead.html	★☆☆☆☆
4	Hackerfactor.com	https://www.hackerfactor.com/blog/index.php?/archives/881-Kim-Jong-Undead.html	★☆☆☆☆
5	Hackerfactor.com	https://www.hackerfactor.com/blog/index.php?/archives/881-Kim-Jong-Undead.html	★☆☆☆☆

Figure 3.5: Output displayed with relative information, when user clicks on ”submit” button underneath tweet URL text box

Chapter 4

Technologies Used

Here are the Technologies used in order to implement our web app, and how we ensured they worked with the system.

- HTML/CSS + Bootstrap 4
  - For website navigation bar, Tweet URL text box, ”submit” button, and over-all functionality/styling.
- Photoshop
  - Creation of necessary graphics displayed.
  - Must ensure that all graphics created are able to be displayed using our HTML/CSS, and alongside the Bootstrap 4 graphics.
- Python3 + TensorFlow + SKLearn
  - TensorFlow and SKLearn libraries will be referenced by Python3 code, and will aid with natural language processing and machine learning, utilized to predict a validity score.
  - Python3 will also be used to code a web crawler to retrieve information from the twitter URL, news websites for consideration as sources, given a tweet or content.
  - Need to ensure that our Python3 code is built in a manner that it correctly integrates with the front end of the web-app.
  - Must ensure that web-scrapers built using python3 can correctly scrape information using various types of tweet and page formats.
- Django
  - IDE used for development and local deployment of web-application in python3

    - Fully tested to ensure that web-app contains API to correctly handle data transfer between front-end and back-end Python3 code.

- GitHub
    - Used to pull and push code to make for easier collaboration and reduce probability of data loss

Chapter 5

Architectural Diagram

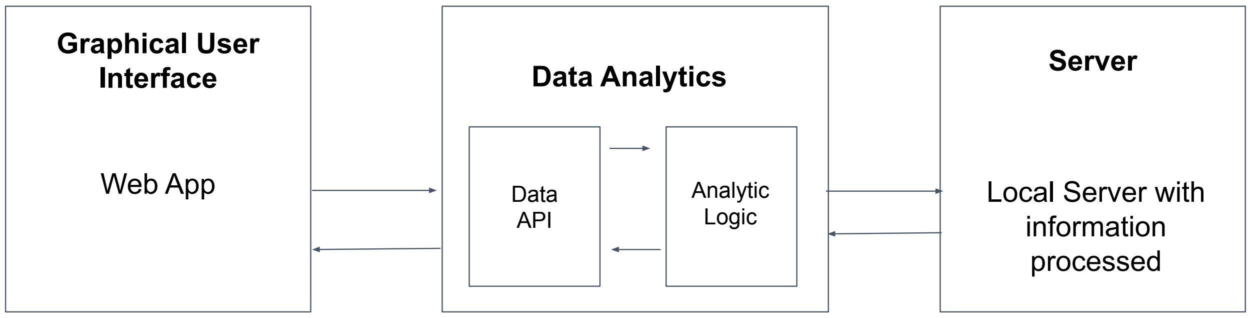


Figure 5.1: The architecture diagram provides a high-level overview of all modules in the software system.

Chapter 6

Design Rationale

6.1 User Interface

Our system is easy to use and simple by design. The main functionality lies in the back end, which is completely abstracted from the user. The user must simply be on the website, and will have access to two major features of the product. The first being the authenticate (submit) button underneath the tweet URL text box, and the second being the information displayed when the authenticate button is clicked. The information displayed contains all relevant data regarding the validity of the content on one page. Given the data heavy feed of social networking sites such as twitter, it is best to have a simple display with information for each individually inputted tweet. Else, given a constant influx of tweets/information, the user will be lost.

6.2 Technologies Used

For the back-end data processing, we used python3 alongside Tensor Flow and SkLearn to aid with natural language processing and machine learning, utilized to predict a validity score. Additionally, we used a web-crawler coded in python3 to retrieve information from twitter. A news API (newsapi.org) was used to retrieve the content of potentially relevant news articles for consideration as sources, given a tweet or content. Furthermore, we utilized a local host machine to run our back-end Machine Learning code. The API itself was coded in python3, and sends information back and forth between our front-end and back-end system. This ensures that no intensive processing is being from the users side. The actual submit (authenticate) button was added using HTML/CSS given the bootstrap library. Lastly, we used GitHub for version control as it makes it easier to work collaboratively and thus allowing us to finish the project faster.

Chapter 7

Implementation

7.1 Tweet Reliability Score

The tweet reliability score consists of the following: **Cosine Similarity Score * Article Score**

7.1.1 Cosine Similarity Pre-processing

1. Data Crawl tweet from URL submitted by User.
   - Retrieve information including content of tweet.
2. Remove all stop words + unknown symbols from content of tweet.
   - Words such as ”to, from, the, a” whose meanings do not change relative to the context they are placed in. These words can be considered as noise and are removed from the string.
3. Use RAKE NLTK library to extract keywords from String
   - Rapid Automatic Keyword Extraction: Uses frequency of word appearance, co-occurrence with other words in text, and collocations.
   - Collocations are the probability that one word appears after another word. For example, there are known phrases that appear in everyday language and have particular meanings. With this in mind, a probability is computed that demonstrates how likely a word fits in a phrase with another.
   - Example
     - *Text: ”It is confirmed that Xinyu, Kyle, and Hetesh can detect fake news”*
     - *Result: [(’detect fake news’, 9.0), (’confirmed’, 1.0), (’xinyu’, 1.0), (’kyle’, 1.0), (’hetesh’, 1.0)]*
   - ”Fake” and ”News” are more likely to appear after the word ”detect”, than ”confirmed”, ”Xinyu”, Kyle”, and ”Hetesh” are. Thus, there is a weightage of 9.0 placed in the initial phrase.

4. Iterate through News API and Google Fact-Check API with keyword as query
    - Store results of news articles that are outputted when a returned keyword from the RAKE NLTK methodology is placed as a query.
5. Search for combination of extracted keywords that yield highest cosine similarity result between article, and tweet.
    - Iterate through each article result outputted relative to the keyword, and compute the cosine similarity of the tweet String, with the content of the article.
    - Combine different variations of single words in the keywords array, and compute the cosine similarity of the result outputted with the tweet string, with them as a query.

7.1.2 Cosine Similarity

1. Turn Article and tweet into TFIDF vector
    - TFIDF: Term Frequency Inverse Document Frequency

    $$Tf - idf(w) = tf(w) * idf(w)$$

    TFIDF is an algorithm used to transform text into a weightage representation. Returns a high value for a term, if it occurs many times in a document, but occurs fewer times in another.
    - Term Frequency:
        - The more times a single word appears, the higher it's value is relative to the other words. However, in this case there is a higher probability that our article will contain a given word more times than our tweet (As a tweet is limited to 280 characters, and an article can potentially be of infinite length). Thus, to normalize the occurrence of a word with the size of the full content (document), the term-frequency is computed as:

        $$tf(w) = doc.count(w)/TotalWordsInDoc$$
    - Inverse Document Frequency
        - We must consider words that are very common across all documents, and therefore although they have a high occurrence count, they contribute minuscule value. To account for this, the Inverse Document Frequency is computed as the following:

        $$idf(w) = log(TotalNumberOfDocuments/NumberOfDocumentsContainingWord)$$
2. Computer Cosine Similarity given document vector

- The formula for cosine similarity is as given (With D1 and D2 being the tweet and article):

$$CosSim(d1, d2) = DotProduct(d1, d2)/||d1|| * ||d2||$$

  Cosine Similarity measures the cosine of the angle between two vectors, which are projected in a multi-dimensional space. Each dimension corresponds to a single word, and the cosine of the angle between two phrases plotted in those dimensions are measured.

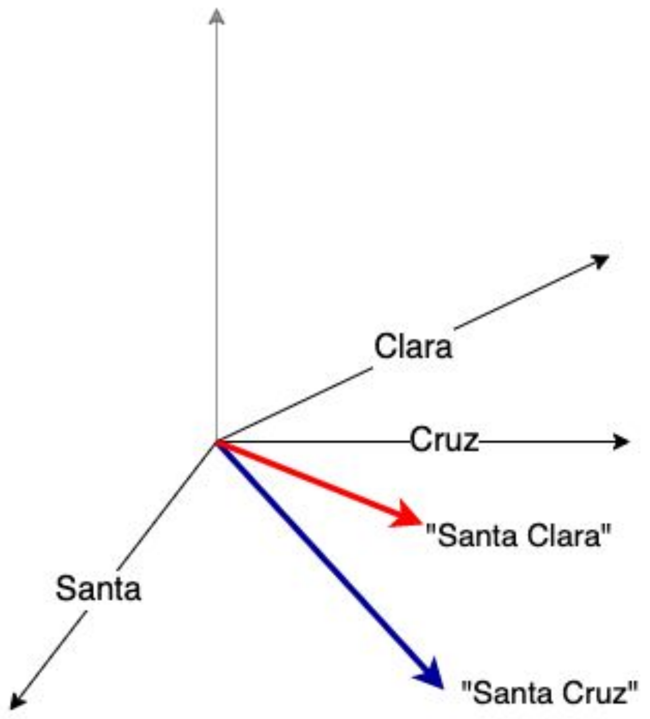


Figure 7.1: Take the two tweets ”Santa Clara” and ”Santa Cruz” as an example. The dimensions of each individual word are plotted. Notice how the dimensions of ”Clara” and ”Cruz” are closer together than that of ”Santa”. The vector value of the two phrases are plotted alongside those dimensions in blue and red. The cosine of the angle between the red and blue plot, is indicative of their similarity.

7.1.3 Article Score

1. Data set

   - FakeNewsNet

     – This is a fake news data repository FakeNewsNet, which contains two comprehensive datasets, which are political and gossip news, that includes news content, social context, and dynamic information.

    - 836,053 lines of data is collected. Each line of data consists of features like news title, news body, labels, retweet number, and so on.

2. Text Preprocessing

    - Text Tokenization

        - I use Keras tokenizer, which is pretrain tool to tokenize the news article. It basically turns to a complete sentence to a set of tokens. For example, Santa Clara University has a beautiful campus. Each of the word from that sentence with be treat as an individual term. Thinking news articles are like rows of words In the end, every word that has been found in all the news article will become a column for each row. If the news article has certain word, that column will be label as one. Later on, we may further process the numerical data by normalize it.

    - Padding

        - Because news articles have different length, which are very hard to build a model to train a dataset like this.So I pad the news articles into same size, basically cutting off the ones are long and the extent the short ones. The padding size is 2986, It choose that is because more that 95.5% of the news article is less that this length. Longer padding size cost longer training time.

3. Build and Training the Convolutional Neural Network

    - Text embedding

        - The article text are passed into separate embedding layers where each word is converted into a 50-dimensional vector.

    - Convolutional layers

        - The resulting matrices from the embedding layer are passed into convolutional layers where 256 filters are applied with kernel size of 10.
        - Due to capacity of automatic learning neural network, each of the filter will focus on single feature of the data, and try to catch the feature after the sample data is passed through.
        - Kernel size means that how many number of dimension will treated together as a unit to catch certain feature.

    - Optimizer and loss function

        - I use the adam optimizer with initial learning rate of 0.01 to optimize the neural network performance

    - The loss function is binary cross entropy. The loss function is formula that measure how close we are from our prediction to the truth. In this case, larger the value is close to 1, the close that the article is true.

4. Model Evaluation

    - Precision Score
        - Highest Precision score 95%
        - Average Precision score 87%
    - F1 Score
        - Average Precision score 86.1%

7.2 Account Rating

1. Data Collection

    - In order to accurately rate the twitter profiles, and build a machine learning classification mode, many sample data are needed for the training and testing. A total of 10,000 data points were collected from online social media profile rating databases such as Social Blade, Sideqik, Brandwatch, Revluence, and Influencer Marketing Hub, all to be used for our model.
    - There are a total of 6 categories of focus, or in ML terms, features.

        (a) Following and Followers Count
        - A low following and high follower ratio, is a good indicator that the account is legitimate. Else, if the user has an extremely high following count, but a lower follower count, it is a typical sign of botting.

        (b) Total Tweet Count
        - Too low of a total tweet count indicates the account rarely posts or is inactive, and too high of a total tweet count can reveal automation behavior.

        (c) Account Age
        - Brand new accounts have a high potential to be modified or botted.

        (d) Average Retweets and Likes
        - Engagement percentage is calculated using the average retweets and likes. A high Retweet/Like count yields a high engagement count, thus improving the overall account rating.

2. Various Prediction Models

    - For our account rating systems, we use various prediction models to test for the most effective solution. The first being K-means, also known as K means clustering, is an effective method that picks various seeds based on our data set. It is an unsupervised method that is capable of separating the accounts info based on the features, and sorts them with a rating.

3. Final Prediction Models

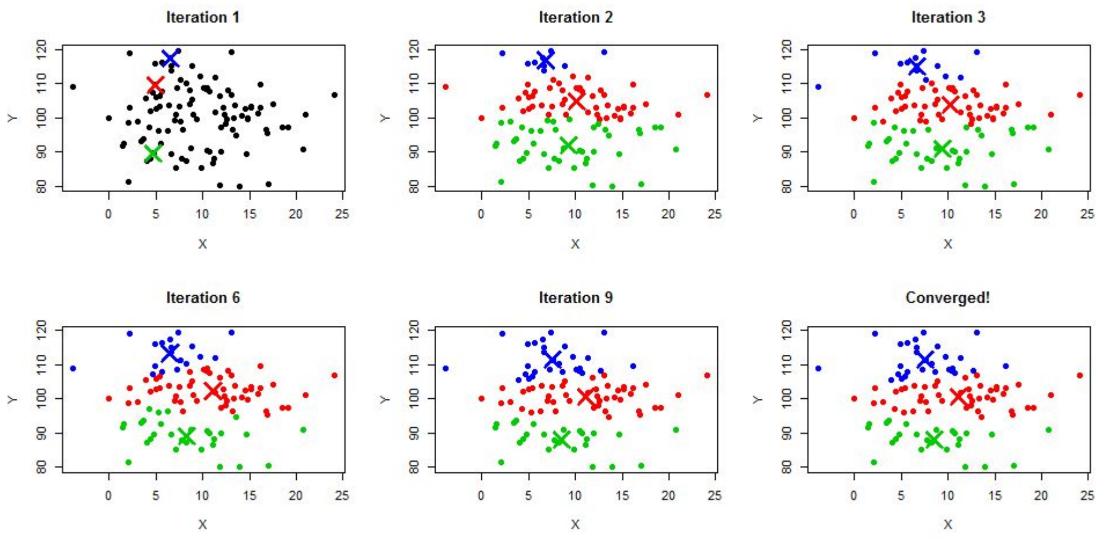
    - Next Linear Discriminant Analysis and Quadratic Discriminant Analysis are incorporated using SKLearn.
    - LDA
        - As seen from the LDA diagrams beneath, LDA produces a linear line that separates the clusters in our data. This effectively classifies each portion into its own category. In order to test the accuracy of the classification system, we take 80 percent of the data points as training data, and used the remaining 20 percent to test against our ground truth. For LDA, we were able to reach an accuracy of 78.6 percent.
    - QDA
        - As seen from the QDA diagrams beneath, QDA creates a quadratic separation between clusters and separates them by rating. As one compares the QDA and LDA graphs on the bottom, it is evident that QDA is more accurate than LDA as it can adjust into non-linear shapes. When tested against our ground truth, we were able to achieve an accuracy of 92.4 percent, which was compared against the ratings from the databases.

4. Data Query from Account Rating

    - As shown

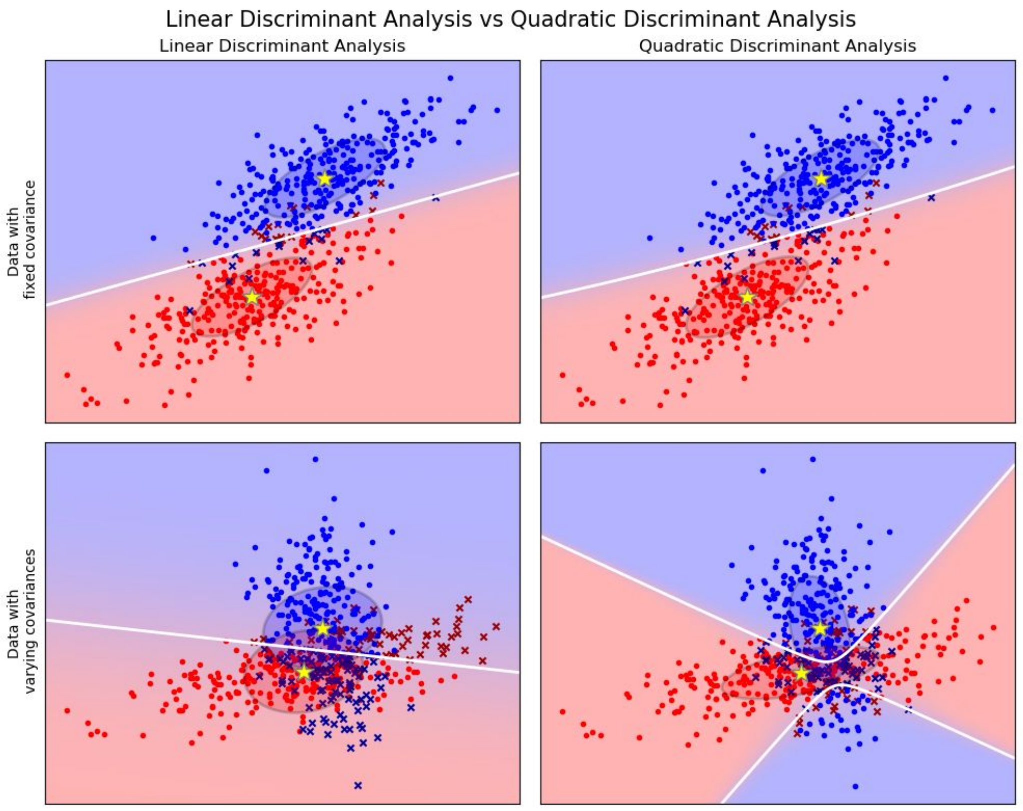
7.3 External Fact Check

Using the exact same process as generating keywords from tweets as described in the Cosine Similarity Pre-Processing phase, enter and parse results from the Google News API.



```
k–means ['A++' 'A++' 'A++' 'A++' 'A+' 'A+' 'A+' 'A+' 'A+' 'A+' 'A+' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'B–' 'B' 'B–' 'B–' 'B–' 'B–' 'B' 'B' 'B'
 'B' 'B–' 'B' 'B–' 'B–' 'B–' 'B–' 'B' 'B–' 'B–' 'B–' 'B' 'B' 'B–' 'B–' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B–' 'B' 'B' 'B' 'B–' 'B–' 'B–' 'B'
 'B' 'B–' 'B–' 'B' 'B' 'B' 'B–' 'B' 'B' 'B' 'B' 'B' 'B–' 'B' 'B+' 'B' 'B'
 'C+' 'B' 'B' 'B' 'A' 'B' 'B' 'B' 'B' 'A' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'A' 'A' 'A' 'B' 'A' 'B–' 'B' 'B' 'B–' 'A' 'B' 'C+' 'B+' 'B' 'A' 'A' 'A'
 'B' 'B' 'B' 'A']
```

Figure 7.2: Using K-Means, 6 features, 35 cycles. Classification by clustering by ratings



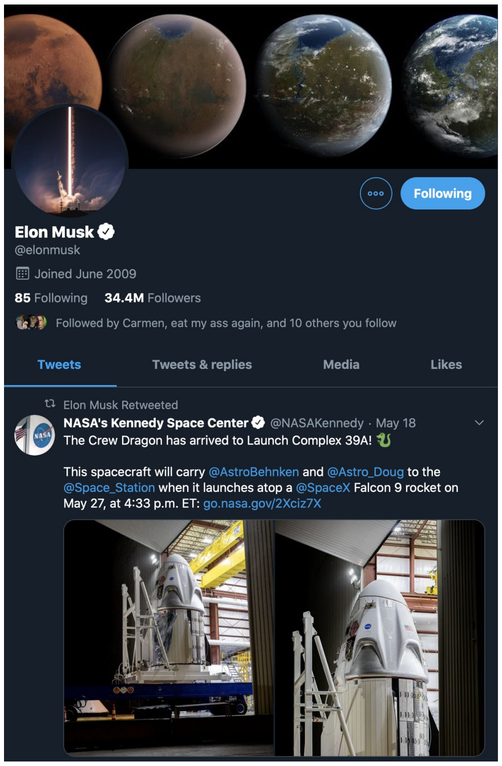
```
lda accuracy: 0.7861635220125787
qda accuracy: 0.9245283018867925

lda ['A++' 'A++' 'A++' 'A+' 'A+' 'A+' 'A+' 'A+' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'B' 'B' 'B-' 'B' 'B-'
 'B' 'B-' 'B-' 'C+' 'B-' 'B' 'B-' 'B' 'B-' 'B-' 'B-' 'B' 'B' 'B' 'B' 'B-'
 'B' 'B-' 'B-' 'B' 'B-' 'B-' 'B-' 'B' 'B' 'B' 'B-' 'B-' 'B' 'B-' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'A' 'A' 'B' 'B' 'B' 'A' 'B' 'B' 'B'
 'B' 'B' 'B+' 'A']

qda ['A+' 'A++' 'A+' 'A+' 'A+' 'A+' 'A+' 'A+' 'A+' 'A+' 'A+' 'A+' 'A+' 'A' 'A'
 'A+' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'B' 'B' 'B-' 'B' 'B-'
 'B+' 'B-' 'B-' 'C+' 'C+' 'C+' 'C+' 'B-' 'C+' 'C+' 'C+' 'B-' 'B' 'B-' 'B'
 'B-' 'B' 'B-' 'B-' 'C+' 'C+' 'C+' 'B' 'B+' 'B' 'B' 'B-' 'B-' 'C+' 'C+'
 'B-' 'C+' 'C+' 'B' 'B-' 'B' 'B+' 'B' 'B+' 'B' 'B' 'A' 'A-' 'B-' 'B+' 'C+'
 'A' 'B' 'B-' 'B+' 'B+' 'B+' 'B+' 'A']

test data: [[709, 49298, 980, 3.59, 1746.48, 21.08]]
lda predicion: ['B']
qda predicion: ['B-']
```

Figure 7.3: Prediction by value, then converting to rating



Elon Musk
@elonmusk
Joined June 2009
85 Following 34.4M Followers
Followed by Carmen, eat my ass again, and 10 others you follow

Elon Musk Retweeted
NASA's Kennedy Space Center @NASAKennedy · May 18
The Crew Dragon has arrived to Launch Complex 39A!

This spacecraft will carry @AstroBehnken and @Astro_Doug to the @Space_Station when it launches atop a @SpaceX Falcon 9 rocket on May 27, at 4:33 p.m. ET: go.nasa.gov/2Xciz7X

```
account: @elavegan

primary account stats:                  qda accuracy: 0.9245
        post count: 1,231
        followers: 969,588              calculated score: 0.8946
        following: 125
        engagement rate: 1.76%  __      rating: B+
        avg like: 1,6671.4
        avg comment: 353.8
```

Figure 7.4: Data Packet that includes all attributes used for full account accuracy detection system

Chapter 8

Testing

The test plan for our project consisted of three major phases: unit testing for each major component, system testing that tests our whole solution together, and acceptance testing that ensures our system fulfills the proposed requirements. Ultimately, our test plan covered all possible cases where errors might pop up and determine whether our implementation is complete.

8.1 Unit Testing

For our unit testing phase, we tested every major and minor module on its own and ensure it works before we bring it into the whole system. For example, one major component is the Machine Learning code used to generate a validity score. For this component, we tested the output given various different input styles. This is important because on the internet, individuals can enter in an almost infinite amount of ascii and alpha numeric characters. We needed to ensure that a correct output is given, relative to the inputted information.

8.2 Acceptance Testing

The last part of our testing ensured that our system fulfilled the requirements that we initially proposed. Here we looked at each requirement, checked our system, and concluded whether we were able to successfully and safely fulfill those requirements. Seeing that we have, we can then conclude that our system works as intended.

Chapter 9

Difficulties Encountered and Lessons Learned

This project demonstrated the true biases behind search queries, and how even a singular word difference can impact the results yielded by news websites. The following are difficulties we encountered, and lessons we learned:

1. Finalizing a method to extract news articles given select keywords
   (a) In some instances, although RAKE NLTK methodology works well, a keyword is simply a compressed meaning of a phrase. Therefore, some context behind the full string is potentially lost.
2. Determining cosine similarity threshold at which an article is considered to support a tweet
   (a) What percentage of similarity should an article be considered to ”support” or ”debunk” a tweet? Should the similarity percentage adjust given a news topic where not much information is known? As of now, we have a fixed similarity threshold that has been adjusted many times through trial and error to account for both articles that support and debunk a tweet. In the future a Machine Learning model could be used to try to maximize/minimize a percentage value for a threshold, relative to the types of articles fed in.
3. Determining how much information to provide to a viewer, to determine tweet authenticity.
4. How to compute the tweet reliability score.
   (a) As of now, tweet reliability score is the product of the article score and cosine similarity score.
5. Readers' biases are subjective and difficult to compute.
6. Training data for fake news is difficult to acquire, as they're mostly unlabeled or removed.
7. Fact-checking is difficult to achieve using Machine Learning, as a news article can be altered in different ways.
8. Accurate keyword selections are crucial in displaying desired results.

Chapter 10

Future Work

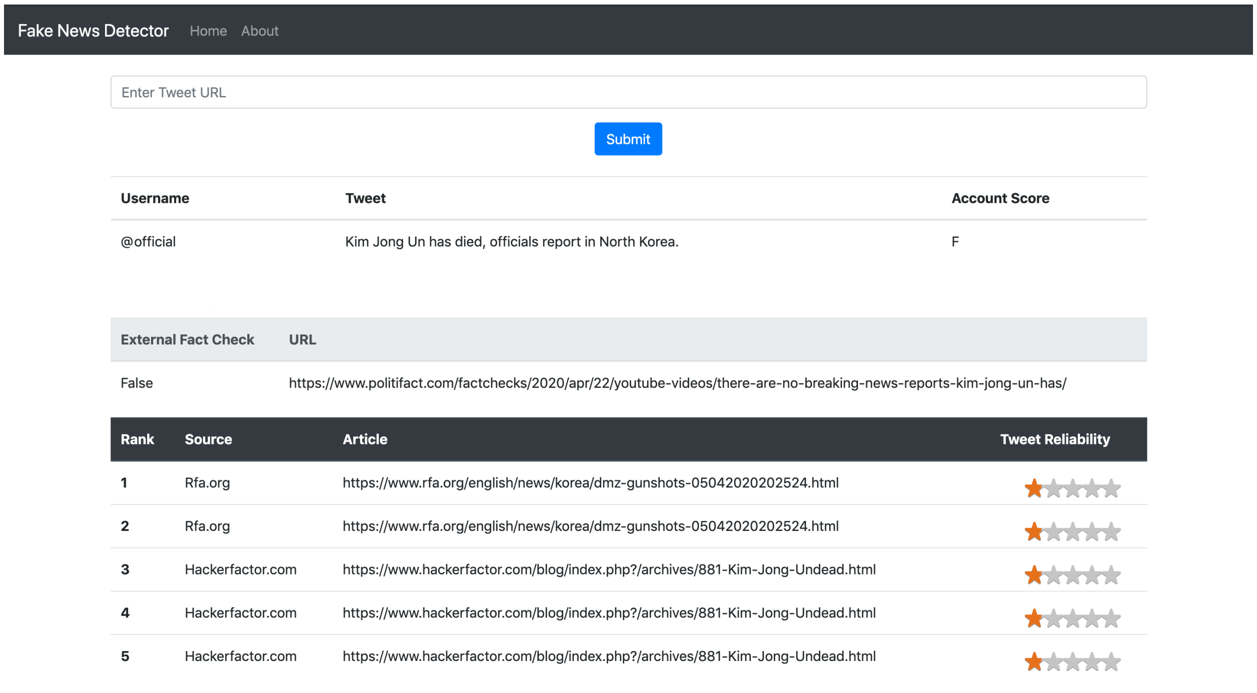
1. Optimize similarity threshold by implementing machine learning model to minimize or maximize threshold, relative to the type of article that is fed in.
2. Add more articles to neural network to potentially improve accuracy.
3. Create web-scraper to automate sourcing of training data for neural network.
4. Add compatible web browser plugin that performs same functionality, however displays results on user's twitter feed.
   (a) Can also push further to allow plugin to display recommendation real-time while user is on a fake or real news site
5. Continually update web scraping methodology as twitter constantly changes their tags and fields. An alternate would be to register using the twitter API.
6. Expand project from just focusing on news, to successfully detecting spam/bot accounts.
7. Add social feature for user rating/submissions, to also take into account general population sentiment towards a statement.
8. Create mobile version of Fake News detection, as a twitter client app.
9. Expand fake news detection to Facebook and Instagram URLs.

Appendix A

User and Installation Guide

A.1 Installation and Setup

1. Install Pycharm from the following website: https://www.jetbrains.com/pycharm/download/
2. For access to the back-end data processing code base, please contact either hsehgal@scu.edu, kjiang@scu.edu, or xchen6@scu.edu for the GitHub link.
3. Once granted access, download and unzip file from Github link.
4. To access the codebase for the front-end and web-server, download the file at: *https://drive.google.com/file/d/1PSMgpZcjhedBk1TI-EfebHslz4gIZMrl/view?usp=sharing*
5. In PyCharm, open the web-app code by unzipping and opening the ”FakeNews” directory.
6. On terminal, use pip to install venv by using the command ”pip3 install venv”
7. Once venv is installed, activate the environment by navigating to the venv folder, and then the bin directory.
8. Once in the bin directory, install all dependencies by running ”pip install -r requirements.txt”.
9. Once in the bin directory, install all dependencies by running ”pip install -r requirements.txt”.
10. Once all requirements are installed, activate the venv by running the activate file in the bin directory.
11. In the back-end data processing code, in ”brain.py”, replace the two API key’s for the google API and news-api with your own valid API’s by creating an account on *newsapi.org* and *https://developers.google.com/fact-check/tools/api*
12. In the FakeNews Web App opened in Pycharm, navigate to the mysite folder, and run the following command ”Python3 manage.py runserver”
13. The website will be online and fully functional at web address: 127.0.0.1



Username	Tweet	Account Score
@official	Kim Jong Un has died, officials report in North Korea.	F

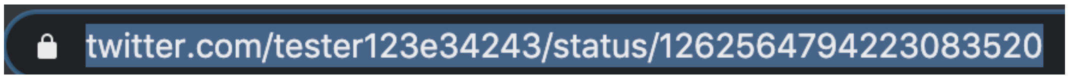
External Fact Check	URL
False	https://www.politifact.com/factchecks/2020/apr/22/youtube-videos/there-are-no-breaking-news-reports-kim-jong-un-has/

Rank	Source	Article	Tweet Reliability
1	Rfa.org	https://www.rfa.org/english/news/korea/dmz-gunshots-05042020202524.html	★☆☆☆☆
2	Rfa.org	https://www.rfa.org/english/news/korea/dmz-gunshots-05042020202524.html	★☆☆☆☆
3	Hackerfactor.com	https://www.hackerfactor.com/blog/index.php?/archives/881-Kim-Jong-Undead.html	★☆☆☆☆
4	Hackerfactor.com	https://www.hackerfactor.com/blog/index.php?/archives/881-Kim-Jong-Undead.html	★☆☆☆☆
5	Hackerfactor.com	https://www.hackerfactor.com/blog/index.php?/archives/881-Kim-Jong-Undead.html	★☆☆☆☆

Figure A.1: Web Page Access containing Tweet Reliability, Author, and External Fact Check scores

The following information is relative to the example show cased in the figure above. The user has entered in a twitter tweet URL in the text box, and the relative results are displayed.

A.2 Entering in twitter URL



twitter.com/tester123e34243/status/1262564794223083520

Figure A.2: Twitter Tweet URL

A user simply copies the twitter URL of a tweet they want to fact check, and paste it on the text box that reads ”Enter Tweet URL” on the web app.

A.3 Reading Account Score

The Account Score is beneath the tweet URL text box, and we can see that the author has received an ”F” grade. This implies that the account mimics that of a bot or newly created account, and therefore has less of a reason to be trusted.

A.4 Reading Tweet Reliability Score

The tweet reliability score is accessible underneath the text header, in the form of a star based ranking system. In this case, we notice that there is a 1/5 star ranking, insinuating that even the most relevant news article sources to the tweet does not fully support it. Therefore, one may be inclined to assume that the tweet is untrustworthy.

A.5 Reading External Fact Check

The external fact check rating is beneath the account score rating, and reads as ”False” with a relative news URL claiming to debunk the tweet.