Virginia Tech
Blacksburg, VA 24061
CS 4624 – Multimedia, Hypertext, and Information Access
Spring 2019

Final Report

Online Role-related User Classification on Twitter

Gregory Pickett, Kenneth Worden, Adam Wilborn
"The Bluebirds"

Client: Liuqing Li
Professor: Edward Alan Fox

May 8, 2019

List of Figures

Executive Summary / Abstract

The main goal of this project is to provide a web application that will host the existing TWIROLE model. “TWIROLE, a hybrid model for role-related user classification on Twitter, which detects male-related, female-related, and brand-related (i.e., organization or institution) users. TWIROLE leverages features from tweet contents, user profiles, and profile images, and then applies the hybrid model to identify a user's role[1].” The main use of TWIROLE is to aid future evaluation efforts and research studies relative to investigations that rely upon self-labeled datasets.

The web application is made to be easy to use and navigate, allowing for a wide range of audiences, including researchers or common Twitter users. Other goals of the project were to add a new classifier to the model that will improve the accuracy of TWIROLE. The model previously had only one advanced feature that used the *k*-top words method to analyze users and classify them. Essentially, looking at all of a user's tweets and ranking the *k*-top words used, classifying a user based on what the words were. The final model includes the previously mentioned advanced feature and an additional advanced feature that analyzes *k*-top emojis similarly to the *k*-top words feature. Once features were added, the model was then trained again on the existing data set improving the accuracy.

The website is made up of an HTML page using React on the front end. The backend (TWIROLE) is made using Django to render the HTML page, host images and other resources and expose a GraphQL API. The front end makes AJAX calls to the GraphQL API which obtains the information that will be displayed on the website. The website is aimed at being simple to manage and update with experience in web development specifically Django, React and GraphQL. The website is hosted by DLIB or the Digital Library Research Laboratory using their dlib.vt.edu domain name [2].

Introduction

Twitter is used by millions daily with 500 million tweets being sent every day, which is equivalent to 6,000 tweets every second [3]. With this many people using Twitter, there is a clear incentive to classify users helping both academic and industrial research. Using accurate and appropriate user classification, one could market specifically to users based on how they tweet.

TWIROLE detects whether a user is a male, female or brand by looking at tweet contents, user profiles, and profile images. To do this, TWIROLE uses a "hybrid model" consisting of three components: "basic features" classifier, "advanced features" classifier, and a convolutional neural network (CNN). The goal of this project is to create a user-friendly web app that will host TWIROLE, similar to Botometer (https://botometer.iuni.iu.edu/), and to also improve the accuracy of the model.

When creating the website we made sure to design it with the users in mind taking a simplistic approach and drawing our design from the app Botometer. As you can see below in Figure 1 the website simply prompts the user to enter their screen name, to which Botometer will output whether the user is a bot or not. Similarly our website will prompt the user for their screen name, which then will be used to display the classification of the user. Our website will also display the level of accuracy using a stacked bar chart.

Once the website was created we sought to improve the accuracy of the model by adding additional classification features. When deciding on which feature to add we tried various methods, testing the accuracy of the model with each potential feature.

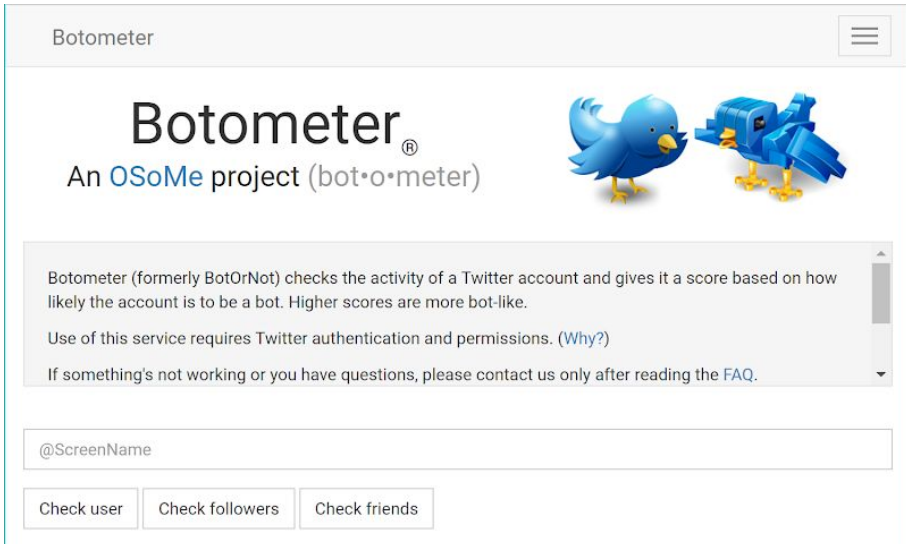

Figure 1: The Botometer website

Objective

The title of this project is *Online Role-related User Classification on Twitter,* or *Twitter Role Classification* for short. The main goal of this project was to make an easily-accessible interface to expose the TWIROLE classification model to others interested in role classification. Rather than implore aspiring hobbyists, researchers, and laymen to install dependencies, environments, and programs, we designed a web application that runs on the Virginia Tech internal network.

The TWIROLE model was already built before we began the project. It achieved an accuracy of 89.72% [1]. In addition to building the website we worked closely with Liuqing Li to add features to the model that would further improve the accuracy of the model.

Client

Our client is Liuqing Li, a Virginia Tech Ph.D. student in the Digital Library Research Laboratory (DLRL). Li is also the manager of the DLRL 20 node Hadoop Cluster, which is a distributed framework/environment for CS4624, CS4984, CS5984, CS5604 and CS6604 at VT. Li's research mainly focuses on Information Retrieval (IR), Tweet Analysis and Text Summarization.

Constraints

Constraints for this project in order to meet the objectives include creating a simple and easily manageable website. To ensure that the website is easily manageable, great documentation will also be provided, so that future contributors can keep up with the website.

We worked with Liuqing to determine the features of the web interface and add model improvements. For the web interface, we were free to choose whatever web technology suited our needs. In this case, we chose to use Python Django due to (1) our client's familiarity with Python, (2) the framework's built-in testing structure, and (3) the ease of development. The web application exposes a GraphQL API which can be used without the need of a frontend, thus allowing others to classify Twitter usernames in other applications.

Requirements

The requirements for this project were determined by an initial meeting with our client, Liuqing Li, in which we discussed goals and stretch goals of the project. From the goals highlighted by our client, the requirements were broken into two sections, website development and model improvement. A stretch goal of having functionality with the Arabic language was also introduced, but was unachievable due to time constraints.

Website development

When designing the website the client expressed clear interest in having a website that looked similar to Botometer as seen in Figure 1. This means that the client wants an application that will take in a username as input and output its predicted class label. The website will be hosted by the Digital Library Research Laboratory using their dlib.vt.edu domain [2].

Model improvement

Changes to the model revolve around improving the accuracy of the model. This meant that we had to try various different features and test which features yielded the greatest accuracy. When making these changes, the client advised us to look into analyzing user names and also using emojis tweeted to classify. We decided to test the accuracy of four different features and the combination of these said features which are explained below.

Potential features to add to model

1. K-top Emojis: This feature looks at the most popular emojis for male, female and brand and uses this to classify users
2. Emoji Frequency: This feature predicts that females use emojis more than men and brands and classifies users based on how many emojis they use
3. Second-person score: This feature calculates a score based on the frequency of second-person terms (e.g. ‘you’, ‘yours’)
4. Third-person score: This feature Calculates a score based on the frequency of third-person terms (e.g. ‘he’, ‘she’, ‘it’, ‘they’’)

Method	Accuracy
Baseline	89.72%
Second + third person scores	89.68%
k-top emojis + second + third person scores	89.75%
k-top emojis	**89.95%**
Emoji frequency score	89.83%
Emoji frequency score + k-top emojis	89.80%

Figure 2: Results from testing features

After testing the potential features we found that the k-top emojis feature yielded the highest accuracy and this feature was added to the model as an advanced feature.

Design

Model Design

Basic Features (BF)

The TWIROLE model was already defined by our client when we started working. TWIROLE employs a "hybrid model approach", a unique approach in the Twitter user classification realm [1]. This approach combines three classifiers with different features in a layered fashion which are then fed as input into the final "hybrid" classifier.

The three classifiers are:

- Basic features (BF)
- Advanced features (AF)
- CNN

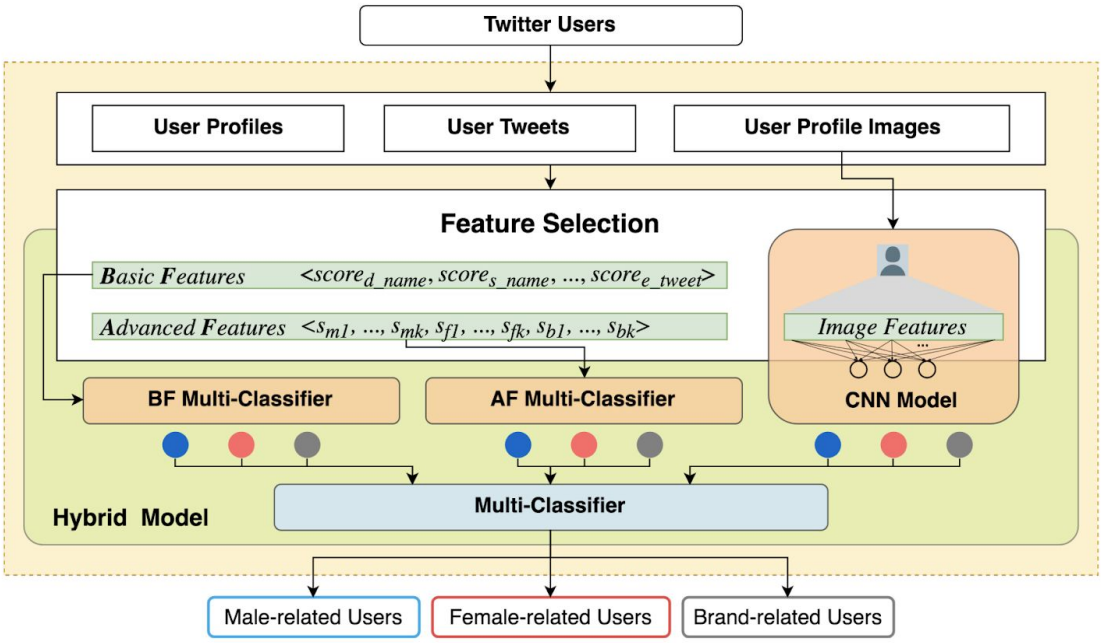

Figure 3: The original TWIROLE classification model [1]

The BF classifier has 5 components, each which contributes to the final score:

1. Display name score
2. Description

3. Follower-to-friend ratio
4. Brightness of profile image
5. Tweet scores

The display name score is calculated by breaking the display name (e.g. @username) into words and calculating the words' frequencies from the babynames dataset [4].

The description score looks at the usages of first-person terms and brand terms, and calculates a score, -1, 0, and 1 where -1 contains brand terms and no first-person terms, 1 contains first-person terms and no brand terms, and 0 is none of the above.

The follower-to-friend ratio is calculated as:

$$score_{tff} = \log \frac{Num_{followers}{}^2 + 1}{Num_{friends} + 1} \qquad (3)$$

Figure 4: The follower-to-friend ratio

The brightness of the profile image looks at the profile image in HSV and gets the average brightness from the entire image.

Tweet scores are calculated by three separate features: first-person scores, interjection scores, and emotion scores. The first-person scores are calculated by counting the number of tweets with first-person terms divided by the total tweets. The rest of the scores are calculated similarly.

$$score_{fp\_tweet} = \frac{\text{\# of tweets that have terms in } list_{first}}{\text{\# of tweets in user tweet collection}} \qquad (4)$$

Figure 5: The first-person score calculation

Advanced Features (AF)

The first advanced feature in the TWIROLE model is the k-top words feature. By default, k is 20. This works by analyzing the entire training dataset and calculates a frequency vector 60 words (20 x 3) long. This word vector is then iterated, where for each word, the number of tweets is counted that contain said word. This sum is divided by the total number of user tweets to calculate the k-top score as shown:

$$score_{k\_top} = < s_{m1}, s_{m2}, \ldots, s_{mk}, s_{f1}, s_{f2}, \ldots, s_{fk}, s_{b1}, s_{b2}, \ldots, s_{bk} > \qquad (5)$$

Figure 6: How the k-top score vector is calculated for a user

The second advanced feature in the TWIROLE model is the one we added, the k-top emojis feature. This feature works similarly to the k-top words feature however emojis are used instead of words in this instance.

```
training_male_screen_name = benchmark.screen_name[training_male_index]
training_female_screen_name = benchmark.screen_name[training_female_index]
training_brand_screen_name = benchmark.screen_name[training_brand_index]

ktop_emojis_dict = sc.get_ktop_emojis(training_male_screen_name,
                                      training_female_screen_name,
                                      training_brand_screen_name,
                                      'benchmark_1/tweets_clean', 10)
```

```
[(':+1:', 1304), (':fire:', 247), (':rofl:', 177), (':large_blue_circl
e:', 131), (':white_circle:', 126), (':punch:', 112), (':bear:', 103),
(':squid:', 102), (':comet:', 99), (':man_shrugging:', 96)]
[(':sob:', 16389), (':heart_eyes:', 14282), (':joy:', 10243), (':kissing_
heart:', 7534), (':blush:', 4782), (':weary:', 4010), (':two_hearts:', 28
84), (':purple_heart:', 2825), (':roll_eyes:', 2115), (':tada:', 1902)]
[(':rotating_light:', 114), (':point_right:', 99), (':spider:', 97), (':s
tudio_microphone:', 76), (':rocket:', 76), (':minidisc:', 67), (':shoppin
g:', 67), (':india:', 65), (':jp:', 41), (':tickets:', 36)]
```

Figure 7. Emojis most commonly used by each class (first array male, second array female and third array brand)

Convolutional Neural Network (CNN)

The CNN feature employs a pre-trained CNN, ResNet-18, which takes the entire user image and outputs to 3 classifications which are interpreted as male, female, and brand [5]. ResNet-18 was modified to have 3 output tensors, in order to suit the needs of TWIROLE [5].

Website Design

Frontend

For the website, we used React for view templating and Redux for application-state management. It was designed as a single-page application. Figure 8 details an initial sketch for the web application.

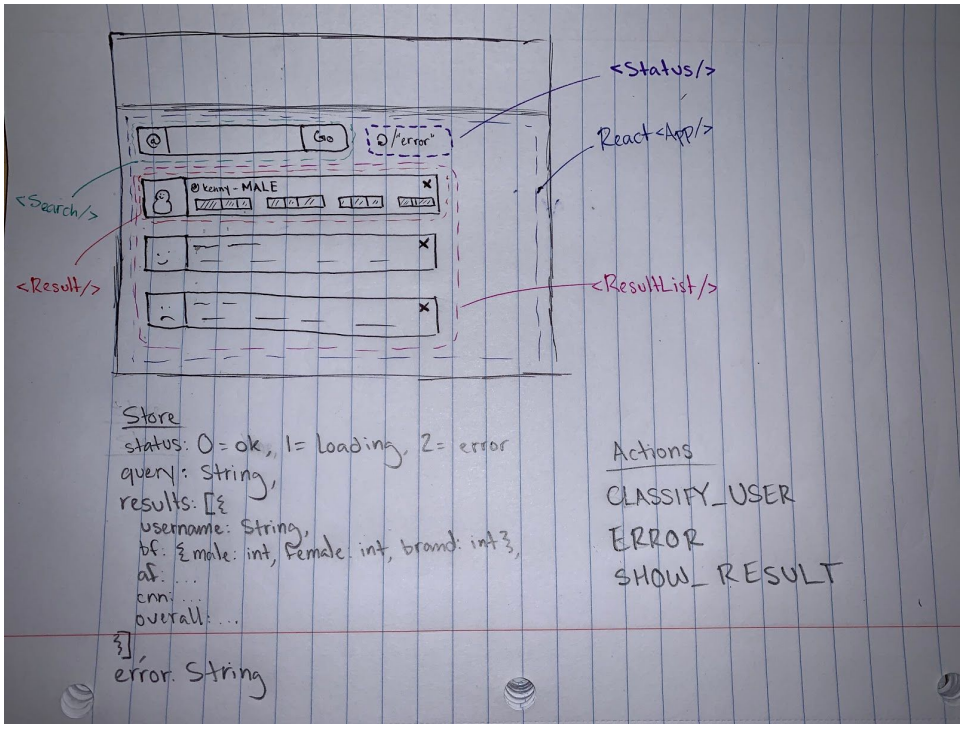

Figure 8: The website interface design

The implementation closely follows this initial design. As you can see, the design in Figure 8 breaks down each element of the website into its component parts, a Search component that houses the input field and button, a Status component that displays a spinner/error message depending on the status of the application. A ResultList component houses individual results. The entire application is conventionally housed inside of a App component.

Below the UI is a sketch of the application state in JSON format. The status of the application is one of three (3) values: 0 (OK), 1 (LOADING), and 2 (ERROR). Another notable field in the application state is the results array, which is an array of JSON objects describing the classification results for a particular user.

Testing/Evaluation/Assessment

For testing, we apply the 10-fold cross-validation method. This means a sample is randomly split into 10 equal sized subsamples. Out of these 10 subsamples, one subsample is kept to test the model and the other 9 are used for the training. During training, TwiRole calculates the feature scores from BF1 to BF5 as the input. It then trains the BF multi-classifier with our role-related labels. For each user, the output is a probability vector of male, female, and brand. After that, TwiRole trains the AF multi-classifier with the k-top words score vectors in the same way as the BF multi-classifier. We are currently adding k-top emoji score vectors to further improve the AF multi-classifier. Following that, the profile images of all the training users and their respective labels are put into the ResNet-18 model to train the deep neural network [5]. Finally, we combine the three probability vectors and train the final multi-classifier.

Users' Manual

Model

TwiRole: A Hybrid Model for Role-related User Classification on Twitter
We publish a pre-trained version of TwiRole for role-related user classification on Twitter. The model can automatically crawl a user's profile, profile image, and recent tweets, and classify a Twitter user into Brand, Female, or Male, which is an aid to user-related research on Twitter.

Getting Started
Prerequisites
Install Python 2.7 (Anaconda recommended)
Installation
Clone TwiRole's repo on your local machine using:

```
git clone https://github.com/liuqingli/TwiRole.git
```

Next, install essential libraries:

```
cd TwiRole
pip install -r requirements.txt
```

After this, you need to install two NLTK packages in Python 2.7:

```
>>> import nltk
>>> nltk.download('stopwords')
>>> nltk.download('wordnet')
```

Finally, set up your Twarc API key and secret:

```
twarc configure
consumer key: ***
consumer secret: ***
```

Please log into Twitter and visit this URL in your browser:

https://api.Twitter.com/oauth/authorize?oauth_token=***
After you have authorized the application please enter the displayed PIN: ***

First Classification Task
TwiRole can detect a single user or multiple users. The screen names of users should be saved in a CSV file line by line. The output contains the final label and the probability of each role.

In the first run, a PyTorch model will be automatically downloaded. User files will be saved in "./user"

To classify a single user use:

```
python user_classifier.py -u [screen_name]
```

To classify multiple users use:

```
python user_classifier.py -f [CSV File]
```

Website

This section details the operation of the TWIROLE web application. The website is a single-page application (SPA) which loses its state when the page is refreshed. Static links to the source code and TWIROLE paper are seen at the top-right of the page. The design is purposefully kept simple, the only actions being (1) username classification and (2) result deletion.

The application has a search bar where the user can input a Twitter username and run the classifier. An overview of the idle application is seen in Figure 9.

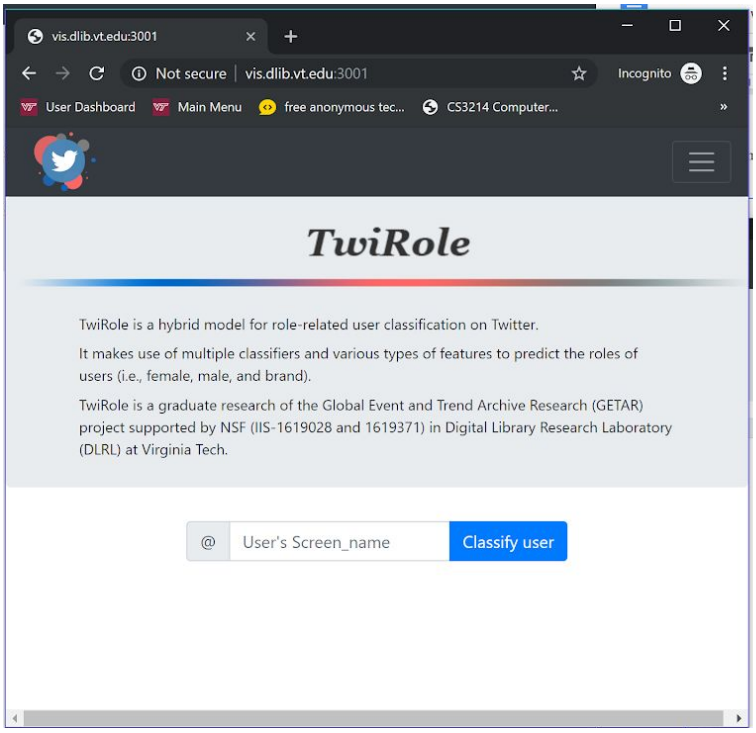

Figure 9: The TWIROLE website, as it appears upon launch

Classification is a two (2) step process. First, the user inputs a Twitter username in the search bar. Then, they click “Classify user”. This is seen in Figure 10.

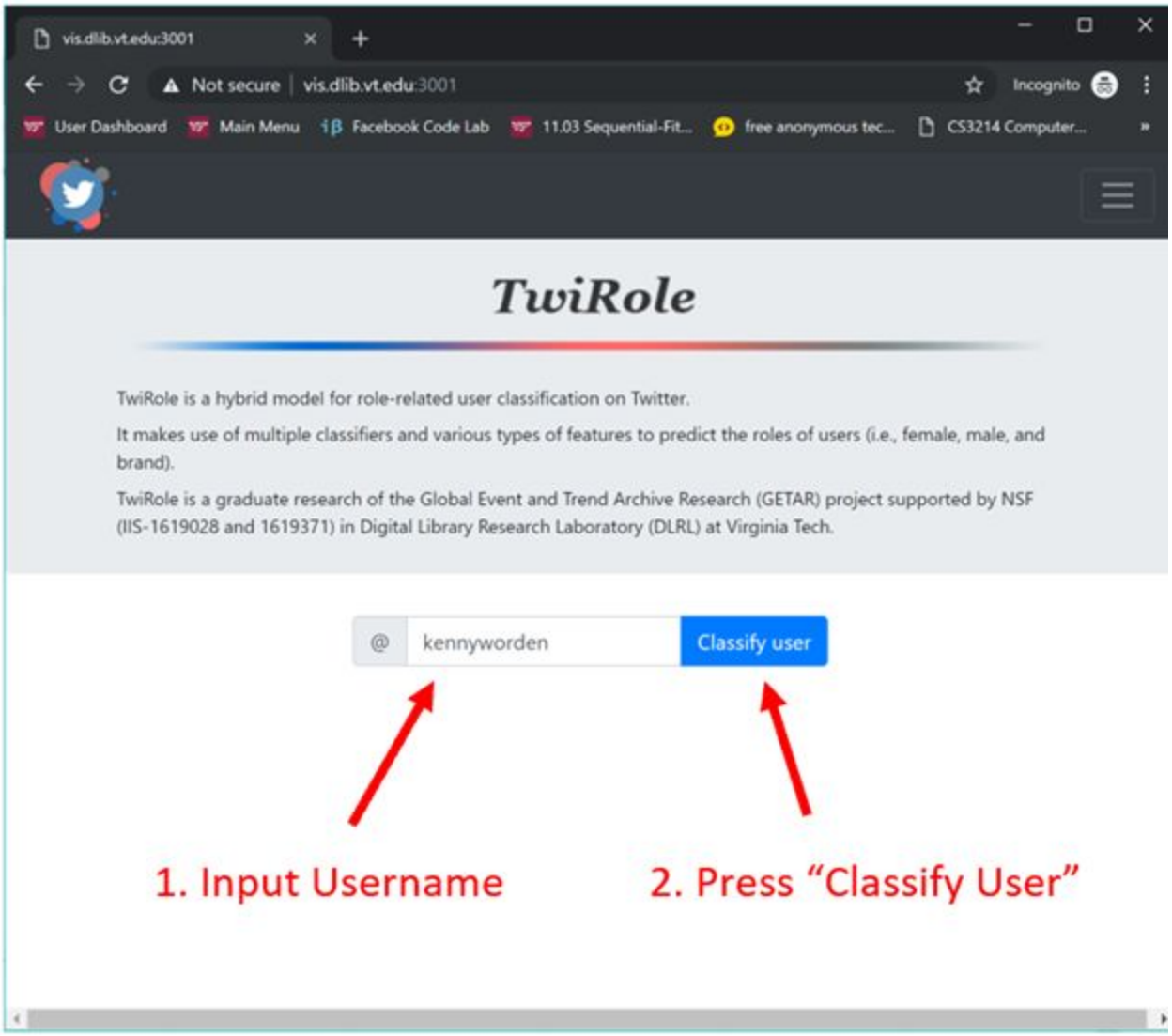

Figure 10: How to operate the TWIROLE website

The user will then see a spinner as the classifier processes the inputted username. During this time, the search bar and button are also disabled. This can be seen in Figure 11.

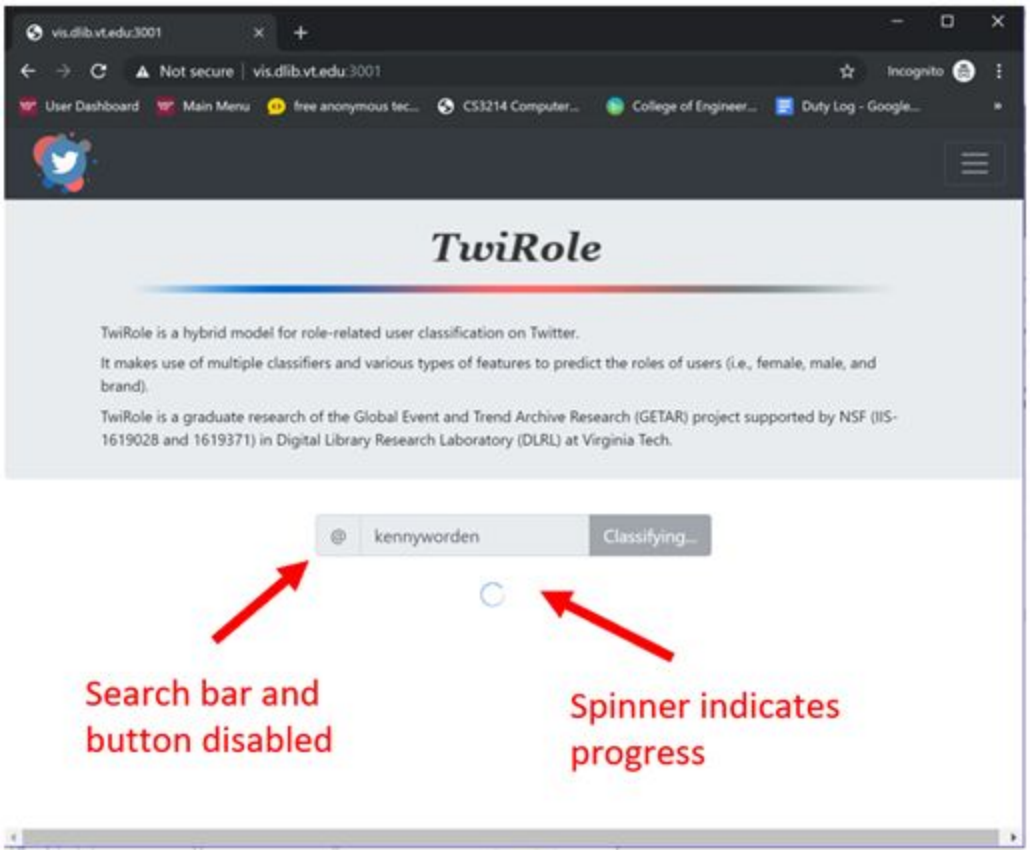

Figure 11: An illustration of what the web application looks like during classification

From here, one of two things can happen. In one case, the entered username was successfully classified, in which case classification results appear below the search bar. This is seen in Figure 12.

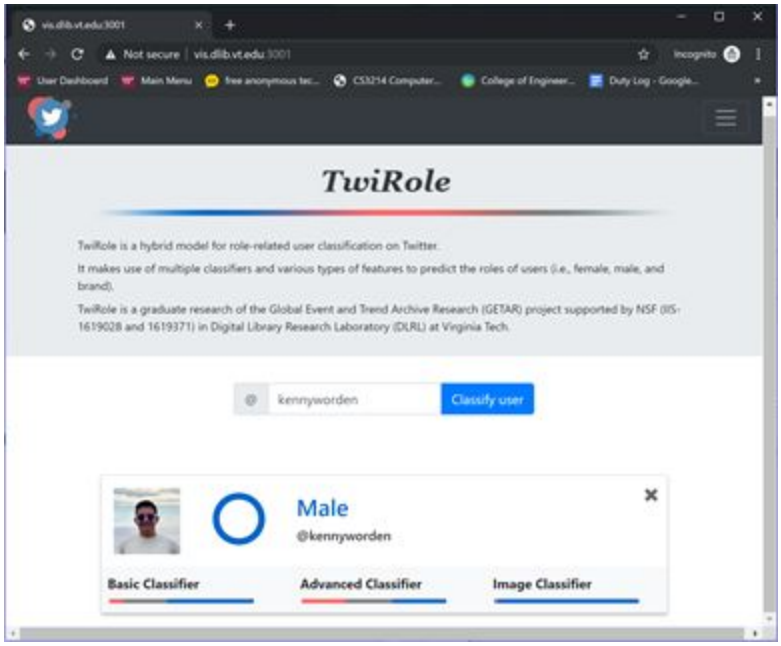

Figure 12: The result of a successful classification

The user can click the “X” button in the upper-righthand corner of any result to remove it from the web application. This action is final.

On the otherhand, attempting to classify a non-existent Twitter username, or causing internal classification problems will lead to application error. This is handled gracefully by the web application, as seen in Figure 13.

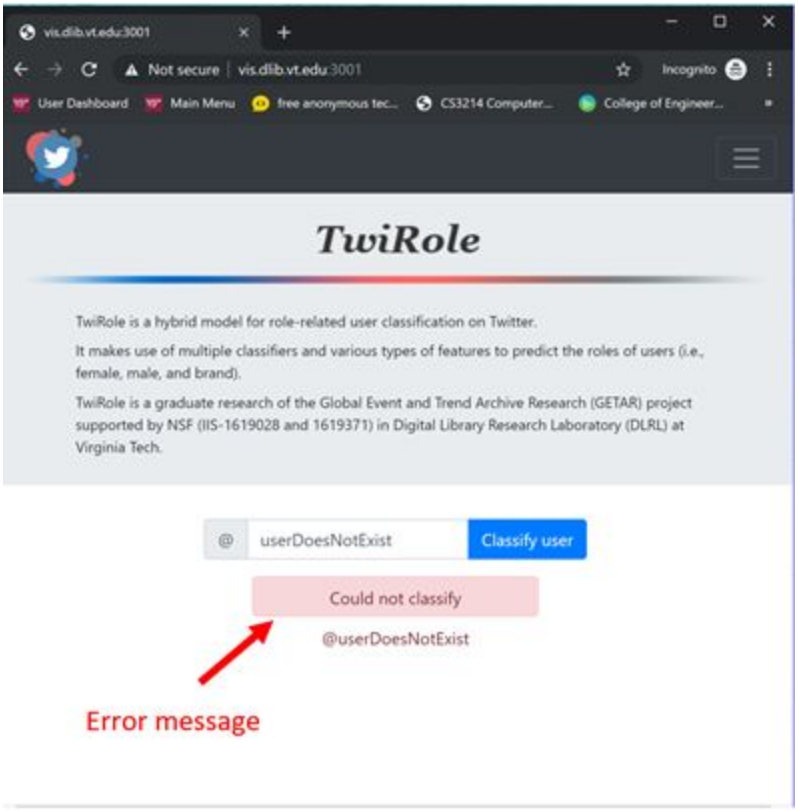

Figure 13: An illustration of an errored classification

The error message is very general. The most common source of errors is:

1. The Twitter user does not exist
2. Lack of internet connection
3. An internal classification problem

The first error can be rectified by checking the username for spelling mistakes or any mistyping. The second can be addressed by making sure the appropriate cables are attached, and that the device being used to access the website has a working internet connection.

User Evaluation

In order to ensure that our website achieves its intended functionality and provides a good user experience we proceeded with a user evaluation. Our user evaluation involves of a set of tasks pertaining to our website that we will give test subjects. The test subject's interaction will be evaluated based on time taken to complete the task and transparency of tasks being completed.

First user

Task:

1. Classify yourself using your Twitter @name
2. Classify multiple users
3. Classify a user that doesn't exist

Task 1 Classify yourself using your Twitter @name

Name of user: Bella Filardi
Steps of task

1. Enter your Twitter name into the search bar
2. Click the button to classify user
3. Read results and interpret

Time taken: 1 minute
User feedback: Simple to understand what is going on. There is only one source of input. Easy to see what the classification percentage is.

Task 2 Classify multiple users

Name of user: Bella Filardi
Steps of task

1. Enter your Twitter name into search bar
2. Click the button to classify user
3. Enter the Twitter name "@Pilajes" into the search bar
4. Click the button classify user
5. Read results and interpret

Time taken: 1 minute
User feedback: Simple to perform, but would be nice if the field cleared after a successful classification.

Task 3 Classify a user that doesn't exist

Name of user: Bella Filardi
Steps of task

1. Enter the Twitter name "@yah" into the search bar
2. Click the button classify user
3. Read results and interpret

Time taken: 1 minute
User feedback: The error is a bit general, but she understood that the website had trouble classifying the given user.

Second user

Task 1 Classify yourself using your Twitter @name

Name of user: Sergio Pozo
Steps of task

1. Enter your Twitter name into search bar
2. Click the button classify user
3. Read results and interpret

Time taken: 1 min
User feedback: Interface is self-explanatory and easy to use, yet I'm not sure what each feature means.

Task 2 Classify multiple users

Name of user: Sergio Pozo
Steps of task

1. Enter your Twitter name into search bar
2. Click the button classify user
3. Enter the Twitter name "@Pilajes" into the search bar
4. Click the button classify user

5. Read results and interpret

Time taken: 2 min
User feedback: Not clear that typing in an additional user will list more users

Task 3 Classify a user who doesn't exist

Name of user: Sergio Pozo
Steps of task

1. Enter the Twitter name "@yah" into the search bar
2. Click the button classify user
3. Read results and interpret

Time taken: 1 min
User feedback: Good error handling

Third user

Task 1 Classify yourself using your Twitter @name

Name of user: Nabil Saad
Steps of task

1. Enter your Twitter name into search bar
2. Click the button classify user
3. Read results and interpret

Time taken: Less than 1 min
User feedback: Intuitive interface with website only having one possible task to be completed

Task 2 Classify multiple users

Name of user: Nabil Saad
Steps of task

1. Enter your Twitter name into search bar
2. Click the button classify user
3. Enter the Twitter name "@Pilajes" into the search bar
4. Click the button classify user
5. Read results and interpret

Time taken: Less than 1 min
User feedback: Great that multiple users can show up at once

Task 3 Classify a user that doesn't exist

Name of user: Nabil Saad
Steps of task

1. Enter the Twitter name “@yah” into the search bar
2. Click the button classify user
3. Read results and interpret

Time taken: Less than 1 min
User feedback: Simple error handling, only recommendation is more CSS to polish page

Developer's Manual

Model

To add new features to the model, you first need to edit the _score_calculator.py file and add any new features you would like. For instance, if you look at the file, we have added a ktop_emojis_score function that calculates the ktop emojis score for the model. After you added a new feature to that file, you need to update the Jupyter notebook called Evaluation_1_cross.ipynb. You will need to add code to call your new feature under the section labeled "For each iteration." For instance, you can see where we added calls to the ktop_emojis_score function in this section. After you have properly added the code for this, you will then need to add additional code in one of the "training and testing" sections depending on if your new feature is for the basic or advanced features. In the case of the k-top emojis score, we had to add it to the "training and testing" section for the advanced features. After you have added the additional code needed to train and test your feature, simply run the Jupyter notebook to see if the accuracy has improved or not.

Website

This project is meant to run under pipenv. If you choose not to run under pipenv, you can install the packages as specified in the Pipfile in Conda, or raw via Pip.

Configuration (development)
This project requires a few things to be installed:

- Pipenv
- NPM
- Python 3.7+

Once these are installed, run the following commands from this directory:

```
1. pipenv install
2. npm install
```

In the `project/settings.py` file, there is a Python variable **`TWIROLE_DIR`**. This must point to a valid TwiRole classifier directory on the local file system.

As can be seen in the `api/twirole.py` script, it runs the `./classify.sh` script in the TwiRole classifier. This script should be adjusted to meet your configuration needs. However, this project assumes that `./classify.sh` will correctly classify a user.

Running the server
Once properly configured, you can run the script:

```
./runserver.sh
```

DEBUG and **PORT** are variables configurable from within runserver.sh.

File hierarchy
The project is organized into 5 folders.

api - A Django app that contains code pertaining to the API. Exposes a GraphQL endpoint and contains functions to call the TwiRole model (see: `twirole.py`).

app - The React application. The entry point can be found in `index.jsx`. It uses a Redux store to maintain application state.

assets - The static assets directory. By default, these are served on `/static`. Contains Javascript libraries, CSS, and some images such as the spinner and TwiRole logo.

page - A Django app that contains static page-serving code. This currently only serves one route, as you can see in `views.py`.

project - The project configuration folder. This contains the `settings.py` file, which is used to configure the project.

templates - The HTML template files. Currently, `index.html` is the only file that is served, but it is composed of many templates in partials.

Changes to the backend or frontend will be immediately realized by the runserver shell script. They can be applied in a local web browser by refreshing the page.

Conclusion and Future plans

This project has been very fun to work on and provided us with a visual application of machine learning which was extremely interesting. This project also allowed for us to see how tweets can be used to classify people and in turn potentially subject one to specific marketing. If we were to continue to work on this project we would need a larger data set to improve the accuracy. We received a lot of feedback on how certain people were misclassified because they tweet a certain way leading them to be incorrectly placed into traditional gender roles. For example, in our VTURCS presentation a woman who tweets mostly about her technical career was classified as male. If we were to have a larger dataset we could train the model to notice these subtle differences.

Lessons Learned

1. **Knowing which features to include in the model is difficult**
   As can be read in the TWIROLE paper, discovering which features to use and with which parameters is a process of trial-and-error. When deciding on whether to use k-top emojis and the second-person score, there was really no way to know, without retraining the model, whether we would see an improvement.
2. **Our model will never be perfect**
   Since we are using tweets to classify people into strict categories that not everyone fits into our model will never be perfect. There will always be someone who doesn't agree with our classification and that is completely ok. Being upfront with our users was the best way to appease the discomfort of placing people into categories.
3. **There is no such thing as too much data**
   Looking back on our project the only thing that we could have helped us improve our model is a larger dataset. Which is quite tedious to generate when it comes to tweets.

Acknowledgments

We would like to thank the VT CS department and NSF for its support of the Global Event and Trend Archive Research (GETAR) project through grants IIS-1619028 and 1619371

The team would like to gratefully acknowledge the contributions of our client, Liuqing Li, for his work on the original version of TwiRole. To contact our client, his email is liuqing@vt.edu.

We would also like to thank the students of our class for providing constructive criticism on our project. Lastly, we want to acknowledge our professor, Dr. Fox, who helped us with various aspects of the project.

References

[1] L. Li, Z. Song, X. Zhang, E. A. Fox, *A Hybrid Model for Role-related User Classification on Twitter*. 2018. Available: https://arxiv.org/abs/1811.10202

[2] A. Wilborn, G. Pickett, K. Worden, L. Li, E. A. Fox, *Twirole Internal Website.* 2019. Available: http://vis.dlib.vt.edu:3001/

[3] K. Smith, *58 Incredible and Interesting Twitter Stats and Statistics*. 2019. https://www.brandwatch.com/blog/Twitter-stats-and-statistics/

[4] Kaggle.com, *US Baby Names*. 2017. Available: https://www.kaggle.com/kaggle/us-baby-names

[5] K. He, X. Zhang, S. Ren, J. Sun, *Deep Residual Learning for Image Recognition.* 2015. Available: https://arxiv.org/abs/1512.03385