Tweet Turing Test: Detecting Disinformation on Twitter

John Johnson, Katy Matulay, Justin Minnion, Jared Rubin
`{jmj382, km3868, jm4724, jar624}@drexel.edu`

College of Computing & Informatics, Drexel University, Philadelphia, PA USA

Abstract

Coordinated online disinformation campaigns linked to state-sponsored entities have been used to subvert online discourse and influence public opinion. Declassified evidence, released by the U.S. government to the global intelligence community and reaffirmed by Twitter, show that such disinformation campaigns were in effect during the 2016 U.S. presidential election. Recognition of a tweet's intent, whether posted in an authentic context versus with the intent to "troll" by spreading disinformation, is a challenge which continues to evolve. In this paper, we seek to build a supervised classification machine learning (ML) model that can accurately and efficiently identify potential sources of disinformation on Twitter. We based our dataset on a large and ubiquitous corpus (*n≈2.1e6* tweets) of confirmed troll tweets and balance it by acquiring a separate corpus (*n≈1.5e6* tweets) of randomized tweets by verified Twitter users. We merge these two sets of tweets to one dataset (*n≈3.6e6* tweets in total), conduct exploratory data analysis, and preprocess. We then apply multiple approaches for classification, using 1) transformer-based pre-trained language models based on Google's BERT, 2) conventional ML models with meticulously engineered features based on characteristics identified as indicators of disinformation, and 3) a novel combination of these two models in a transformer-based multimodal model. Our study finds that all three approaches demonstrate high performance metrics (typically 90-99% scores on all metrics tested). We concluded that 1) traditional ML models with sufficiently descriptive engineered features can perform as well or better, and with fewer computing resources, than a deep learning/transformer-based model, but also that 2) even the most basic BERT-based model with inexpensive data preprocessing can achieve 85% or greater predictive accuracy. Code for our project is open-sourced and available through a public GitHub repository.[1]

Keywords: NLP, disinformation, trolls, Twitter, BERT, machine learning, classification, multimodal modeling, transfer learning, deep learning

1. Introduction

1.1. Disinformation

Disinformation is a subtype of strategic field operations, used to subvert online discourse and influence public opinion by intentionally altering the information environment. In the age of social media and online news, online information warfare is both a valuable weapon for a foreign adversary to harness as well as a challenging weapon to defend against.

Disinformation and strategic information operations, when considered in the context of trolling by Russian operatives, have also been described as "astroturfing". Zelenkauskaite writes that "the paid and orchestrated aspect of Russian trolling renders it comparable to astroturfing" with overt similarities to propaganda campaigns in that their aim is to "provide misleading information by altering reality" [1]. Within the context of the 2016 U.S. election, discussed in further detail in the next section, Russian trolls were paid by the Russian state to create chaos online in the social media space [2]. This chaos was accomplished via tactics such as distracting, deflecting attention, and sowing seeds of distrust, all in the aims of negatively influencing public opinion and affecting the outcome of the Presidential election [3].

[1] Our code is available at: `https://github.com/disinfo-detectors/tweet-turing-test/`

1.2. Historical Context (2016 U.S. Election)

According to the declassified 2017 Intelligence Community Assessment distributed by the Director of National Intelligence, it was assessed with a high degree of confidence that "Russian President Vladimir Putin ordered an influence campaign in 2016 aimed at the US presidential election, the consistent goals of which were to undermine public faith in the US democratic process, denigrate Secretary Clinton, and harm her electability and potential presidency" [4]. By harnessing the power of social media, specifically Twitter, the Russian state-sponsored Internet Research Agency (IRA) was able to exert a campaign of chaos, further eroding public trust in media, government, and democracy one tweet at a time.

As part of the Mueller investigation, Twitter released a report that quantified just how far reaching these troll accounts had been—over a million Twitter users had been notified that they had interacted in some way with these accounts [5]. Twitter then officially released a dataset of tweets and accounts from IRA associated "Russian troll" accounts, which was then further analyzed by researchers at Clemson University and finally released by FiveThirtyEight [6] in the public domain on both Kaggle and GitHub. This dataset serves as the control group of labeled "Troll" accounts/tweets for our project.

1.3. Twitter Platform

Twitter is a social network and real-time communication service that allows users to blog short messages of not more than 280 characters [7]. These messages are labeled as "tweets", and can contain emojis, metadata tags, gifs, and hyperlinks. Twitter has become a service for friends, families, and coworkers to connect instantly and share information of their choice. The platform has also become an important outlet to spread news amongst the masses. It has a userbase of roughly 500 million people from all over the world. The social media site has many use cases and has been a data storage warehouse by housing information uploaded by all users.

1.4. Twitter Verification

1.4.1. Legacy Verification Policy

During the time period in which our data originates (the years 2013-2017), Twitter used a verification policy to vet certain user accounts, with the intention of providing other Twitter users with a means of determining the authenticity of an account. A verified account meeting the requirements of the policy would display a blue checkmark icon adjacent to the account's display name.

The verification policy has since changed (refer to the next section for more info on the change), but prior to the change the verification policy was advertised with the following meaning: "The blue Verified badge on Twitter lets people know that an account of public interest is authentic" [8]. The three core requirements for verification are that an account is 1) authentic, 2) notable, and 3) active.

To ensure *authentic* users, the legacy verification requirements called for a photo identification check and similar credentials to prove a user's identity. For a user to be considered *notable*, the account needed to represent a prominently recognized individual or organization. Finally, an *active* account meant the account must have a demonstrated history active use and of adherence to Twitter's rules [9]. The robust legacy verification criteria serves as the control dataset for our project, otherwise referred to as "Verified" or "Authentic" tweets.

1.4.2. Recent Changes

Prior to Twitter's recent acquisition and privatization by Elon Musk on October 27, 2022, a verified account at Twitter had a prescriptive and rigorous definition (as noted in prior section). It is important to note that on November 9, 2022, Twitter stopped accepting applications for a blue verified badge under the criteria used in this research paper, and it is now referred to as "Legacy Verification policy" [9]. We discuss the impact of these recent changes on our study in section 4.1.

1.5. Pre-trained Language Models

Pre-trained Language Models (PLMs) represent a recent advance in the fields of Natural Language Processing (NLP) and in a broader sense in Transfer Learning. By using a pre-trained model "that has already been trained on a related task and reusing it in a new model" [10] we can realize significant reductions to requirements of time, data, and computing resources.

Deep models such as the Transformer [11] have advanced the performance of PLMs to venture beyond context-free word embeddings and instead "capture higher-level concepts in context" [12]. Prominent examples of transformer-based PLMs include Google's BERT [13] and OpenAI's GPT [14].

1.5.1. Transformer Architecture

Transformer models are a specific and novel architecture of neural networks [15]. A key component in a transformer is an *encoder*, a conceptual block which takes inputs in one form and converts ("encodes") them to another form. In the case of PLMs, this encoding process starts with an input embedding step. Inputs, often in the form of a group of words forming a sentence, are mapped (as real-valued vectors) to a word embedding space such that words with similar meanings are mapped more closely than words with dissimilar meanings [15].

This first embedding would not necessarily differentiate words with multiple meanings, such as the word "bank" used in the context of "the bank of a river" or "a bank loan". This is addressed first by an additional encoding step to consider a word's meaning with respect to its position within a sentence, generating a positional embedding intended to represent a word's context. From here, the relevance of a particular word when compared to other words in the sentence is considered. This is accomplished using a technique called self-attention, and an attention vector is generated for each word. When the attention vectors directed at a given word are combined by weighted average, we generate a "final attention vector" for each word [15]. These final attention vectors are sent to a feed-forward neural network to create the final encoded output of a single encoder layer.

Transformer models like BERT use stacks of encoders, with each encoder layer's output being fed as input to its subsequent encoder layer (see Figure 1 for a conceptual diagram of this encoder stack) [15,16]. In the case of BERTBASE, there are twelve encoder layers.

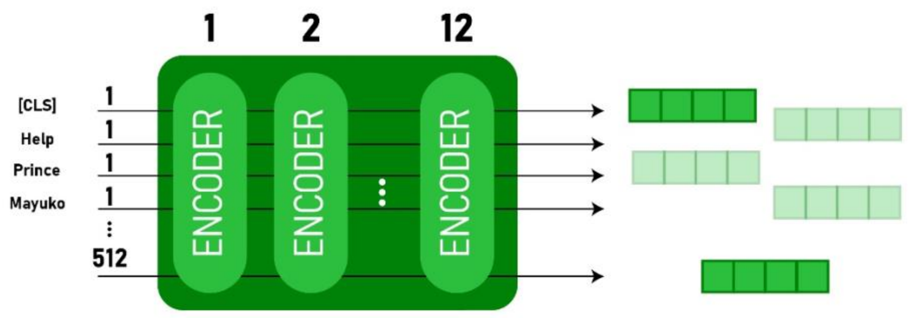


Figure 1 – A conceptual diagram of BERT showing its stacked encoders. Image Source: [16]

For a given large corpus of text, a vocabulary can be extracted, and word embeddings generated. When given a self-supervised training task such as Masked Language Modeling (MLM), and when the large text corpus is sent through the transformer's encoder layers, pre-trained model weights can be determined. These pre-trained model weights effectively represent the mechanics and semantics of the text corpus' language [12,16,17]. Subsequent models can *fine-tune* a PLM for a specific task.

Transformer models like BERT use only encoder layers [17]. Other models, especially generative PLMs like GPT, will contain different/additional components such as *decoders* [15,17], but the discussion of decoders and the models that utilize them is outside the scope of this project.

1.5.2. Hugging Face

Hugging Face is an AI community platform which hosts tools that enable users the ability to build, train, and deploy ML models based on open-source code and technology. Also available to the public are various pre-trained versions of BERT. With the detailed process provided by Hugging Face, our group was able to apply the BERT models and fine-tune them to our selected parameters.

1.6. Objectives

The overall motivation of the project (including analysis and modeling) was to build a supervised classification machine learning model that can accurately and efficiently identify potential sources of disinformation based on fine-tuned BERT models, natural language processing (NLP) algorithms, and machine learning (ML) models, utilizing characteristics identified as indicators of disinformation. The ultimate goal of the ML model is to produce a binary classification of a tweet/account to indicate whether it is a Troll (1) or Authentic (0).

We set out to achieve 3 goals with this project: (1) Fine-tune BERT models for sequence classification tasks to predict and classify solely based on tweet text; (2) Utilize feature engineering, NLP, and traditional ML Models to predict and classify based on the features derived from both tweet text and account metrics, and identify features of importance; (3) Combine both 1 & 2 utilizing the best performing fine-tuned BERT model and tabular features to predict and classify.

We have open-sourced our project code within a public GitHub repository available at:

`https://github.com/disinfo-detectors/tweet-turing-test/`

1.7. Literature Review

Numerous academic and journalistic endeavors have been undertaken to analyze and extract actionable information from this dataset. In a 2016 paper, Bessi and Ferrara estimated that about 400,000 bots engaged approximately 3.8 million tweets about the 2016 U.S. presidential election, accounting for about one-fifth of the entire political discourse [18]. Foundational studies done by Zannettou, Caulfield, and others in the field of disinformation research examined the word frequency, topics, account categories, and derived other useful features and observations. Their recent works, alongside others such as Gianluca Stringhini who co-directs the Security Lab at Boston University, identify crucial findings that Russian trolls are influential at spreading URLs not only on Twitter but also on Reddit and Gab, another social networking website known for having a politically right-wing userbase, and were not influential on 4Chan's */pol/* [19]. The work of Zelenkauskaite et al. in 2021 supports prior work which claims that */pol/* is politicized, and that anti-Semitic and racist rhetoric was prevalent after using named-entity recognition (NER) and keyword analysis [20]. Additionally, Zelenkauskaite et al. found that the focus of */pol/* during the coinciding of the Pittsburgh shooting with the midterm elections was on both anti-Semitic and racist comments associated with the shooting and far-right politics associated with the midterm elections, finding intertwinings of anti-Semitic and racist comments with far-right politics [20]. This background from the communication theory aspect of media could suggest that NER and keyword analysis could provide enriched results for future work in Twitter troll analysis by finding a prevalence of anti-Semitic and racist rhetoric associated with far-right politics.

Interestingly, Zannettou and Caulfield, et al. in 2019 found that in terms of word embeddings, ideologies were akin to pro-Trump for Russian trolls, while Iranian trolls were anti-Trump [19]. Zannettou et al. also found that troll accounts adapt and adopt different identities and behaviors over time, deleting Tweets, changing profile information, as well as using the web interface rather than the app and increasing their own followers as a way of boosting network influence [21]. In 2020, Zannettou et al. proceeded to also find that images were a part of an adapted information warfare operation by Russian trolls on Twitter, in that political imagery was more effective and influentially spread than other forms of imagery [22].

One feature this team initially hypothesized to be a prevalent feature is the use of emoticons/emojis. In 2010, Dresner and Herring emphasized the illocutionary force of emojis as a communicative function in the theory of communication, expanding upon computer-mediated communication (CMC) being enriched by emojis [23]. In 2020, Konrad et al. explore the role of stickers and emojis on Facebook Messenger, further expanding on CMC in that emojis can express emotion, emphasize messages, allow for speed or ease of conversation and understanding, as well as allow for reciprocal exchange and easily relating with a wide audience of receivers using a universal language of sorts [24].

2. Methods

2.1. Dataset Acquisition

2.1.1. Data source #1 – Russian Troll Tweets

We chose to use the FiveThirtyEight dataset of "Russian Troll Tweets" from the IRA accounts, publicly available on GitHub [25]. The original dataset was compiled by Darren Linvell and Patrick Warren of the Clemson University Social Media Listening Center [26]. The GitHub repository contains 13 pre-processed CSV files, with additional data elements added by Clemson University researchers who originally curated the dataset. The dataset contains over 2.9 million tweets from ~2800 Twitter handles, with publication dates spanning ~2013-2017. The total merged CSV is 1.2 Gb. This data source was selected because it contained labeled "Troll" Twitter account data and a rich subset of pre-processed features such as: language, region, followers, following, account type, and full URLs.

2.1.2. Data source #2 – Twitter API

Having been approved for the no-cost Academic Research access level of the Twitter API, we were able to utilize the Twitter Tweet Downloader tool [27] to create and execute custom queries. In doing so, we were able to retrieve bulk exports of tweets as JSON-formatted files. In order to carefully define our control group of verified users, we implemented a detailed methodology for curating tweets. We chose to focus on verified accounts based on Twitter's requirements for verification (refer to section 1.4) and their process to vet accounts for verification [9].

We curated two distinct datasets using the tool: Dataset (2A) Tweets by a small number of manually chosen subset of verified accounts spanning the entire 2013-2017 time period of interest, and (2B) Tweets by a large number of randomly chosen verified accounts during randomized samples of time within the 2013-2017 time period of interest. Lists of twitter handles and quantities of tweets from dataset 2A can be found in the Appendix as **Table 6** and **Table 7**. Dataset 2A contained ~500k tweets, whereas Dataset 2B contained ~1 million.

A requirement of the Twitter API which made the acquisition of Dataset 2B more challenging is the requirement to include at least one "standalone operator" in each query [28]. Examples of a standalone operator would be keyword(s) or hashtag(s). Relevant to our query, this precluded our goal to randomly sample tweets with no specific content posted by any verified author. In order to avoid introducing any of our own biases into the selection of keywords or hashtags, we adopted an approach involving stop words. Stop words are words which appear most frequently in a body of text, such as "the", "a", or "is", and are considered to be "insignificant" to the overall meaning of the text [29]. Rather than attempting to choose an unbiased set of meaningful keywords, we constructed our query's keywords entirely from stop words. Because the stop words are expected to be both the most frequently present words but also effectively meaningless words, we could ensure we are sampling from a wide range of tweets without bias. We use the `nltk` library's `corpus.stopwords` module as our source of stop words.

JSON files were then loaded, normalized, converted to pandas dataframes, merged, and further pre-processed. Further illustration of the data acquisition and pre-processing pipeline can be found in the Appendix as **Figure 21** (page 27) and **Figure 22** (page 28).

2.2. Data Preprocessing

Pre-processing involved an extensive amount of time and work. With the dataset already providing numerical and categorical values, the rest of the features predominantly relied on NLP and feature engineering. The "Tweet" feature as mentioned in section 1.3, was an important component involved in much of the feature engineering done.

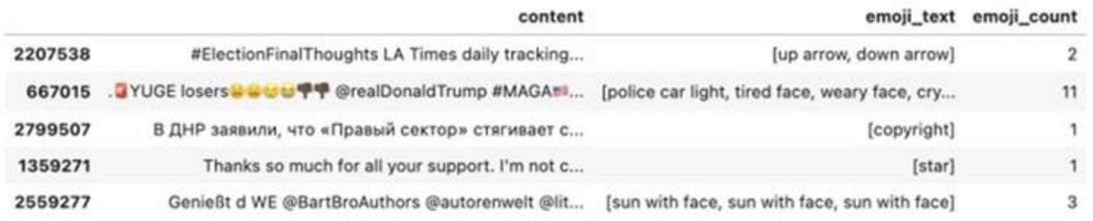
During feature engineering and prior to Pipeline #3, we utilized a tweet cleaning function to align encoding, convert contractions, strip excess spacing and new lines, and remove links and any special characters except hashtags.

2.2.1. English Language subset

The language column contained 30 unique languages, however for the purposes of our work, only English language tweets were retained. The language column on its own was found to be imperfect, as some tweets in Dataset 1 still contained Russian words or other languages despite being labeled as English. The 'account category' column from Dataset 1 was thus used to filter out accounts labeled as "Non-English".

2.2.2. Emoji Preprocessing

In order to extract the text tags for emojis, a function was built using the `demoji` package. Text strings for each emoji were captured in a list and then a count of the emojis per tweet were derived. An example of the resulting columns are shown below in **Figure 2**.



	content	emoji_text	emoji_count
2207538	#ElectionFinalThoughts LA Times daily tracking...	[up arrow, down arrow]	2
667015	.🚨YUGE losers😩😩😩😩👎👎 @realDonaldTrump #MAGA🇺🇸...	[police car light, tired face, weary face, cry...	11
2799507	В ДНР заявили, что «Правый сектор» стягивает с...	[copyright]	1
1359271	Thanks so much for all your support. I'm not c...	[star]	1
2559277	Genießt d WE @BartBroAuthors @autorenwelt @lit...	[sun with face, sun with face, sun with face]	3

Figure 2 – Emoji text and count pre-processing

2.2.3. Tokenization

Open-source Python libraries focused on natural language, such as `NLTK` (Natural Language Toolkit), were used to identify linguistic structures in a tweet and convert them into features. In the `NLTK` package, a module called "TweetTokenizer" has the capability of parsing tweets into tokens. This tool can additionally recognize and preserve retweets, hyperlinks, metadata tags, and gifs.

Originally, the efforts of tokenization had to also be involved on the entirety of our dataset, but with the help of the Hugging Face `tokenizers` library we were able to move our efforts on fine-tuning the best model using an automatic process. Examples of tokenizer outputs are shown in section 3.2.1.

2.3. Exploratory Data Analysis

Exploratory Data Analysis (EDA) was conducted in multiple iterations to learn more about the basic structure of our dataset as well as to probe for insight on potential feature engineering efforts to explore. Results of EDA are provided in section 3.3.

2.4. Feature Engineering

2.4.1. Updates

The updates column existed in Dataset 1 but was manually tabulated for Dataset 2 following the methodology used for Dataset 1. The number of update actions was defined as a sum of the number of retweets, likes, replies, and quotes.

2.4.2. Multi-author

Using the combination of "*external_author_id*" and "*author*", we were able to derive a feature that captured a list of IDs which had more than 1 unique author name, indicating that the user had changed their display or "author" name. The column "*multi_author*" is a binary feature that represents if a given author is in the list of author IDs with more than one unique author name.

2.4.3. Following Ratio

Using the columns "*following*" and "*followers*", the following ratio was derived to show if there was an imbalance specific to troll accounts. To account for users with no followers, the minimum number of followers was set to 1. The equation used to derive this feature and resulting description is below:

$$Following\ Ratio = \frac{Following + 0}{Followers + 1}$$

2.4.4. Emoji Count / Emoji Text

Utilizing the `demoji` python package, we were able to convert UTF-8 encoded emojis to text descriptions, based on the Unicode consortium specification, version 14.0 released on 2021-06-02 [30]. By extracting a list of text descriptions, we then were able to calculate a count of emojis used and one-hot-encode binary features for the 17 most frequently used emojis in the dataset.

2.4.5. Region (Top 15)

The region value from Dataset 1 was determined by a commercial tool from Salesforce called Social Studio, whereas the region value from Dataset 2 came directly from the Twitter API [31]. Based on EDA findings which showed that the top 15 regions represented nearly 75% of the dataset, we settled on one-hot-encoding only these regions and creating a catchall region that represented "All Other". Because there were over 35,000 unique regions in the dataset, this enabled us to utilize this feature without expanding the dimensionality significantly. Any regions that were null were imputed as "Unknown".

2.4.6. Retweet

Because a retweet essentially represents a duplicated tweet, we wanted a binary feature to distinguish whether a tweet was prefaced with "RT", denoting a retweet. Retweets comprised 1/3 of the entire dataset and were retained for fine-tuning and training.

2.4.7. Russian alphabet letter count

Deriving the count of Russian letters in a tweet was based on the hypothesis that non-native English speakers may accidentally insert Cyrillic characters or entire Russian words in a tweet [32]. Using regex, we created a numeric feature to represent the number of Russian letters found in the tweet. The range of letters "А-Я" was input into a lambda function utilizing the Unicode range `'[\u0400-\u04FF]'`[33].

2.4.8. Text-derived features

Using ubiquitous Python packages such as **`NLTK`**, **`regex`**, and **`vaderSentiment`**, we derived several features from the tweet text. These features are described in the sections that follow.

TWEET AND WORD LENGTH

A simple total tweet length was computed on a cleaned tweet. The tweet was first cleaned using **`ftfy`** [34] to align encoding, contractions were converted, excess spacing and new lines were stripped, and then links and any special characters except hashtags were removed. Median word length was also captured as a standalone numeric feature.

PUNCTUATION USAGE

Counts of commas and dashes were extracted into numeric features to determine if native Russian speakers had any discernable punctuation usage patterns compared to verified users.

URL USAGE AND FREQUENCY

The Python package `regex` was used to determine if a URL was present, and if so, how many URLs were present in each tweet. The hypothesis with this feature was that Trolls would have an agenda and would utilize links to external websites to promote their false narrative.

TWEET SENTIMENT / EMOJI SENTIMENT

`vaderSentiment` was used to capture the compound sentiment score for both the tweet text and emoji text. The compound score is computed by summing the valence scores of each word, normalized to between -1 (negative sentiment) to +1 (positive sentiment) [35]. Per `vaderSentiment` reference docs, this is the most useful and commonly used single measurement of sentiment, as it effectively is the weighted composite score.

2.5. Modeling

2.5.1. Pre-trained Language Models Studied

For this study, five PLMs were included. Details of each PLM are provided in their respective papers and are not discussed in detail here. A summary of each PLM is provided below, as well as an explanation of why each PLM was chosen. With the exception of DistilBERT, for each of these models there exists at least a "base" and "large" variant (DistilBERT does not have a variant other than its "base" implementation), with the "large" variant utilizing a larger underlying neural network and significantly more model parameters. For our study, we focused only on the "base" variant of each model.

$\text{BERT}_{\text{BASE}}$ – One of the first pre-trained models released on Hugging Face was the $\text{BERT}_{\text{BASE}}$ model. Trained on 64x Google TPUs, this model took approximately four days to pre-train. Released by Google AI in 2018, the original research team trained the model on BookCorpus and English Wikipedia pages in a self-supervised fashion. The training tasks used to pre-train the model were masked language modeling (MLM) and next sentence prediction (NSP). With a vocabulary size of around 30,000 tokens, $\text{BERT}_{\text{BASE}}$ ultimately contains 110 million parameters [13]. $\text{BERT}_{\text{BASE}}$ was selected because it's a foundational transformer model pre-trained on a large corpus of English-language text.

DistilBERT – The DistilBERT model is a faster, smaller, and cheaper model based on the original $\text{BERT}_{\text{BASE}}$. It has 40% fewer parameters, runs 60% faster, and preserves over 95% of $\text{BERT}_{\text{BASE}}$ performance. The same pre-training dataset as $\text{BERT}_{\text{BASE}}$ was used on this model as well. What makes this model different is the pre-trained objectives set for DistilBERT. DistilBERT maintained the MLM task but added to its objective distillation loss and cosine embedding loss. To successfully train this model, eight Nvidia V100 GPUs were utilized and took roughly 3.75 days [36]. DistilBERT was selected to explore whether it can meet/exceed the performance of $\text{BERT}_{\text{BASE}}$ for our classification task but do so in a more efficient manner.

RoBERTa – Released by Facebook AI in 2019, the team who created RoBERTa asserted that BERT was undertrained and could be optimized. The pre-training dataset used for RoBERTa had about 2.5 TB of filtered CommonCrawl data and contained over 100 languages. The training task involved with this model was MLM. NSP was not included so the model could focus on optimizing the parameters. The computing resources needed to pre-train this model consisted of 1,024 Nvidia V100 GPUs, making RoBERTa the model with the most expensive pre-training cost out of our choices [37]. RoBERTa was selected because it uses a modified approach from BERTbase and claims to be more optimized. We would like to see whether those modifications and optimization can benefit our classification task.

BERTweet – BERTweet is another model released by Facebook AI in 2019 and was the first public large-scale language model for English tweets. The training task included a combination of the $\text{BERT}_{\text{BASE}}$ architecture with the focus on using the RoBERTa pre-training procedure. Additionally, this model was trained on 850 million tweets covering the time frame of 2012 to 2019 [38]. The same GPUs on DistilBERT were also used on BERTweet, but it took 28 days and double the VRAM of each GPU to complete. BERTweet was selected because it mimics RoBERTa but was trained using the same type of data as our dataset.

TwHIN-BERT – A socially-enriched pre-trained model for multilingual tweets is TwHIN-BERT. This model was trained on 7 billion tweets from over 100 distinct languages, consequently consuming the most memory compared to the rest of our models. TwHIN-BERT differs from other pre-trained models as it is trained with not only MLM, but also with social objective. In result of being trained on social engagements, this model can be utilized on not only NLP and recommendation task, but also social recommendation task on Tweet engagement. This model required 16 Nvidia A100 GPUs and approximately 5 days to be a fully functional pre-trained model [39]. TwHIN-BERT was selected because it appeared to have the most promising potential for our classification task, given: 1) its exceptionally large pre-training dataset, 2) the model's pre-training objective included Twitter-specific features involving social engagement on the Twitter platform, and 3) the researchers training the model were directly tied to Twitter itself.

2.5.2. Modeling Pipeline #1 – BERT Models

For our first modeling pipeline, we attempted an approach using only the text of a tweet without any accompanying data. This approach was intended to establish a baseline level of performance for an NLP-focused study. With the recent attention on encoders/decoders and their use in transformer-type pre-trained language models, our group was interested in applying BERT models as the core workhorse of this modeling pipeline. With Hugging Face providing easily accessible BERT-based models and tools to enable the pre-trained models to be fine-tuned, we were enabled to use our dataset with several model types (as explained in the prior section).

To conduct fine-tuning in a controlled environment, a single fine-tuning notebook was created with parameterized inputs wherever possible. An identical corpus of tweets was sampled from the overall dataset for use in fine-tuning. Tweets used for fine-tuning were noted by their unique "tweet_id" to ensure they were filtered out from any dataset used for downstream model training or evaluation (refer to Modeling Pipeline #3). Tweets were separated into training and test sets, with the size ratio of training-to-test as 5-to-1. Both the training and test sets were stratified as 50% troll tweets and 50% authentic tweets, with the selection of tweets within each stratum conducted at random.

To explore the effect on classification performance, multiple sizes of fine-tuning corpora were used with individual performance metrics recorded. To evaluate classification performance, conventional classification metrics such as accuracy, f1-score, precision, recall, area under the receiver operating characteristic curve were calculated. Brier score, discussed in section 2.6, was also calculated. All fine-tuned models generated during this pipeline were set aside for later use in Modeling Pipeline #3.

2.5.3. Modeling Pipeline #2 – Feature Engineering and Non-DL NLP for Tabular Data

To serve as a baseline comparison, three traditional machine learning (ML) models were trained, using the numeric features derived in section 2.4. Logistic Regression, Decision Tree, and XGBoost models were applied to 3 million tweets (stratified on class to a 50:50 split and 80:20 train/test split), with 2.4 million used for training and 600k used for testing. The pipeline was split across 4 Jupyter notebooks, based on feature subset, wherein each notebook the same models, hyperparameters, and evaluation metrics were used. The notebooks and feature subsets were split as follows: a) full 49 features; b) 29 tweet text derived features; c) 20 account features; and d) reduced dimensionality 15 features. The feature subsets are shown in the Appendix as **Table 9**.

The Logistic Regression model coefficient method was used to identify features of importance, while serving as a baseline metric against the more robust XGBoost model. XGBoost was chosen because it is an optimized distributed gradient-boosted decision tree (GBDT) and is notable in that it differs from other gradient boosting techniques due to its regularization mechanism to prevent from overfitting [40,41].

2.5.4. Modeling Pipeline #3 – Multimodal Models

With classification being our end goal, we ran into an obstacle to develop an effective ensemble model that could utilize both our fine-tuned transformer models as well as out tabular engineered feature data. With the `multimodal-transformers` library provided by the fintech firm Georgian [42,43] , it was possible to overcome this obstacle and combine transformers, one hot encoding, and normalization into one single modelling pipeline. This allowed the pipeline to accept both text and tabular information at the same time.

The `multimodal-transformers` library also included implemented subclasses of the same Hugging Face transformer models, allowing us to apply our fine-tuned models. For our final version of this pipeline, we combined the best performing fine-tuned model from Modeling Pipeline #1 with the tabular data features from Modeling Pipeline #2. This final model was then fed tabular data and tokenized text into a neural network to classify if the tweet is a troll (1) or authentic (0) tweet. A simplified diagram showing the basic process flow of our multimodal model pipeline is shown in **Figure 3** below. A list of tabular data fed to the model is below, grouped by category.

(1) ***Text Features:*** tweet text (cleaned)

(2) ***Account-based features***: followers, following, following ratio, multiple authors, and region

(3) ***NLP Text-based features:*** median word length, tweet length, sentiment, emoji count, emoji sentiment, emoji type, # hashtags, # URLs, # commas, # dashes, and # Russian letters

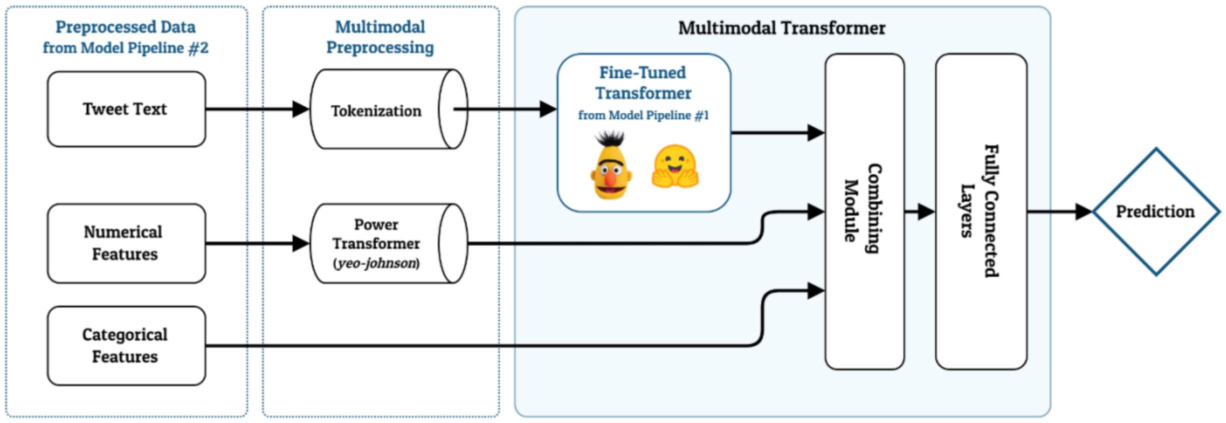


Figure 3 – Diagram of Multimodal Transformer model used in Modeling Pipeline #3.
Based on a similar diagram from the `multimodal-transformer` package [44].

Our approach to training the multimodal model was to select a fine-tuned transformer from Modeling Pipeline #1, select a large subset of tweets (including both cleaned tweet text and tabular data), and filter out of that tweet subset any tweets which had been used for the fine-tuning of the transformer. We then trained the multimodal model in much the same was as we conducted fine tuning in Modeling Pipeline #1 and evaluated using the same predictive classification performance metrics.

2.5.5. Computing Resources

Modeling Pipelines #1 and #3 were trained and tested using a consumer-grade Nvidia GPU with Ampere architecture (RTX 3060 Ti with 8GB VRAM). The computer hosting this GPU utilized an AMD Ryzen 7 5800X (8-core) CPU and 64 GB RAM.

Modeling Pipeline #2 was trained and tested using Google Colab Pro with a Python 3 Google Compute Engine backend, 25.5 GB available RAM and 225.8 GB Disk space, utilizing 0.12 compute units/hour.

Modeling Pipeline #1 was also tested to operate within Google Cloud Platform (GCP). Testing was performed using a GCP Vertex AI Workbench user-managed notebook backed by an `n1-standard-16` (16 vCPUs, 60 GB RAM) compute instance with attached Nvidia Tesla-architecture GPU (T4 with 16GB VRAM). While the GCP resource was confirmed to be capable of executing our code, models trained on this GCP resource were not ultimately used for the project.

2.6. Evaluation of Model Performance

Using the `sklearn.metrics` module of the `scikit-learn` library, all models in Pipeline 2 were evaluated on accuracy, precision, recall, and ROC AUC scores. For Pipelines 1 and 3— F1 and Brier score were also used to evaluate the models. Brier score is a scoring function specialized for tasks predicting "mutually exclusive discrete outcomes or classes" [45]. Because our prediction classes can be considered mutually exclusive, we felt this was an interesting score to include in our study. The specific Brier score implementation used was the module included in the `scikit-learn` library [46].

For a binary classification problem, the proportion of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are important metrics, readily obtained via a confusion matrix. Receiver Operating Characteristic (ROC) is a probability curve that plots the rate of TP against FP at threshold values and demonstrates the performance of the model. Area Under the Curve (AUC) describes the ability of a binary classifier to distinguish between classes and summarizes the ROC curve [47]. The calculations below are the basis for the numbers derived from confusion matrix:

$$Accuracy = (TP + TN) / (TP + FP + TN + FN)$$

$$Precision = TP / (TP + FP)$$

$$Recall = TP / (TP + FN)$$

3. Results

3.1. Data Acquisition

The combined dataset contained 18 features (after initial EDA but prior to post-EDA feature engineering) and 3.6 million records. These preliminary features are listed below. The final data dictionary (after post-EDA feature engineering) can be found in the Appendix as Table 5.

- `external_author_id` (*string*) – a unique identifier tied to a user account, represented as a 64-bit unsigned integer but stored as a string.
- `author` (*string*) – the Twitter handle (a displayed username) of the person posting the tweet
- `content` (*string*) – the text of a Tweet, including #hashtags, @mentions, URLs
- `region` (*string*) – a region classification (e.g. United States)
- `language` (*string*) – the language with which the Tweet is associated (e.g. English)
- `following` (*integer*) – the number of accounts the author handle is following at the tweet time
- `followers` (*integer*) – the number of accounts following the author's handle at the tweet time
- `updates` (*integer*) – the number of update actions on the tweet (see section 2.4.1)
- `post_type` (*string*) – whether the tweet is a retweet or quote-tweet
- `account_category` (*string*) – a general account theme, coded by Linvill and Warren [48]
- `tweet_id` (*string*) – a unique identifier tied to the tweet
- `tco1_step1` (*string*) – for shortened t.co URLs, the first URL to which the first t.co URL redirects
- `data_source` (*string*) – from which dataset the tweet originated (see section 3.3.1)
- `has_url` (*boolean*) – whether or not a URL is present in the tweet text
- `emoji_text` (*list[str]*) – emoji characters converted to English words (see section 2.4.4)
- `emoji_count` (*integer*) – a count of the emoji characters present in the tweet (see section 2.4.4)
- `publish_date` (*datetime*) – the date and time the tweet was sent

3.2. Preprocessing Results

3.2.1. Tokenization

Modeling Pipelines #1 and #3 rely on Hugging Face tokenizers to preprocess tweet text for input to a transformer model. Each of our five PLMs had an associated Hugging Face `tokenizer` subclass, and while there was some overlap between PLMs, each `tokenizer` subclass would ultimately preprocess an identical tweet in different ways. To demonstrate this, we prepared an example tweet containing a variety of Twitter-specific natural language elements. We then fed that same example tweet into each of the five `tokenizer` subclasses. Table 1 shows the results of this test.

Table 1 – Comparison of tokenizer output

Hugging Face Tokenizer Subclass (Associated PLM)	**Tokenized Tweet Text** *(Displayed as a list, converted from token_ids back to vocabulary words)*
None (Raw text)	`This tweet has it all! #Hashtags, @mentions, http://hyperlinks.to/stuff, big words like pauciloquent, proper nouns like George Washington, and even emoji 🤗.`
BertTokenizerFast (BERT$_{\text{BASE}}$)	`['[CLS]', 'this', 't', '##wee', '##t', 'has', 'it', 'all', '!', '#', 'hash', '##tag', '##s', ',', '@', 'mentions', ',', 'http', ':', '/', '/', 'hyper', '##link', '##s', '.', 'to', '/', 'stuff', ',', 'big', 'words', 'like', 'pau', '##ci', '##lo', '##quent', ',', 'proper', 'nouns', 'like', 'george', 'washington', ',', 'and', 'even', 'em', '##oj', '##i', '[UNK]', '.', '[SEP]']`
DistilBertTokenizerFast (DistilBERT)	`['[CLS]', 'this', 't', '##wee', '##t', 'has', 'it', 'all', '!', '#', 'hash', '##tag', '##s', ',', '@', 'mentions', ',', 'http', ':', '/', '/', 'hyper', '##link', '##s', '.', 'to', '/', 'stuff', ',', 'big', 'words', 'like', 'pau', '##ci', '##lo', '##quent', ',', 'proper', 'nouns', 'like', 'george', 'washington', ',', 'and', 'even', 'em', '##oj', '##i', '[UNK]', '.', '[SEP]']`
RobertaTokenizerFast (RoBERTA)	`['<s>', 'This', 'Ġtweet', 'Ġhas', 'Ġit', 'Ġall', '!', 'Ġ#', 'Hash', 'tags', ',', 'Ġ@', 'ment', 'ions', ',', 'Ġhttp', '://', 'hyper', 'links', '.', 'to', '/', 'stuff', ',', 'Ġbig', 'Ġwords', 'Ġlike', 'Ġp', 'au', 'cil', 'oqu', 'ent', ',', 'Ġproper', 'Ġnoun', 's', 'Ġlike', 'ĠGeorge', 'ĠWashington', ',', 'Ġand', 'Ġeven', 'Ġemoji', 'ĠðŁ', '¤', 'Ĺ', '.', '</s>']`
BertweetTokenizer (BERTweet)	`['<s>', 'This', 'tweet', 'has', 'it', 'all@@', '!', '#Ha@@', 'sh@@', 'tag@@', 's@@', ',', '@@@', 'menti@@', 'ons@@', ',', 'http://@@', 'hyper@@', 'link@@', 's.@@', 'to@@', '/@@', 'stuff@@', ',', 'big', 'words', 'like', 'pa@@', 'uc@@', 'i@@', 'lo@@', 'qu@@', 'ent@@', ',', 'proper', 'n@@', 'oun@@', 's', 'like', 'George', 'Washington@@', ',', 'and', 'even', 'emoji', '<unk>', '.', '</s>']`
XLMRobertaTokenizerFast (TwHIN-BERT)	`['<s>', '▁This', '▁tweet', '▁has', '▁it', '▁all', '!', '▁#', 'Ha', 'sh', 'tags', ',', '▁@', 'men', 'tions', ',', '▁http', '://', 'hy', 'per', 'link', 's', '.', 'to', '/', 'st', 'uff', ',', '▁big', '▁words', '▁like', '▁pa', 'uci', 'lo', 'quen', 't', ',', '▁proper', '▁nou', 'ns', '▁like', '▁George', '▁Washington', ',', '▁and', '▁even', '▁e', 'moji', '▁', '🤗', '.', '</s>']`

It's interesting to see how each tokenizer approaches the process of separating this string into tokens. We can see some special tokens being added, e.g. the “[CLS]” and “[SEP]” tokens for BERTBASE and DistilBERT. These tokens are a carryover from BERT's original pre-training where these tokens were used to denote the beginning of a sentence pair ([CLS]) and the separation ([SEP]) between the two sentences of the sentence pair. RoBERTa, BERTweet, and TwHIN-BERT instead adopt an XML-tag-like set of tokens to denote the beginning of sentence (<s>) and end of sentence (</s>).

Other words are broken up into pieces or prepended with special characters. While it was informative for us to see the data being fed to our transformer models, further analysis of the exact mechanism of tokenization and assignment of special tokens for these tokenizers is beyond the scope of this paper.

3.3. EDA Results

Given the high dimensionality of the datasets and importance of contextualizing based on positive/negative class (troll/verified), only multivariate analysis will be highlighted herein.

3.3.1. Data source vs. Account

The data sources were segregated into three categories. The number of accounts within each data source category are visualized **Figure 5**, with details of "verified_user" and "verified_random" categories provided in **Table 6**, **Table 7**, and **Table 8** in the Appendix.

3.3.2. Region vs. Class

The region feature contained both a significant number of null values as well as a significant number of unique values. It's not uncommon for a tongue-in-cheek value to be entered by a user, e.g., NASA's account likes to list "Pale Blue Dot" in reference to the view of Earth from a 1990 photograph taken from space by the Voyager 1 space probe [49].

The Trolls used a much smaller variety of regions in comparison to the verified users—29 vs. 35,444. However, because these values were derived using different methodology it is not a one-to-one comparison. The region feature was retained for use in the models, as 27 unique Troll regions appeared in the Verified dataset; notably absent are the Russian Federation and the Islamic Republic of Iran.

Figure 7 shows a summary of the top 15 most frequently listed regions for each class.

3.3.3. Retweets vs. Class

The binary column "is_retweet" was built from existing labels (for troll tweets from Dataset 1) and tweet object metadata (for verified tweets from Twitter API). The troll dataset had a higher percentage of retweets (see **Figure 6**), but both datasets are more heavily comprised of original tweets.

3.3.4. Bi-grams

With tokenizing the tweets, deeper analysis was done to compare the verbiage of the different twitter users. This included comparing word lengths, tweet lengths, frequent words, and n-grams. **Figure 8** shows the visualization of Bi-grams from a random sample of 100,000 tweets.

3.3.5. Word Cloud

Comparing the words users frequently chose was challenging to display in a digestible way on conventional graphs. A word cloud was created (shown in **Figure 4**) to have a distinct visualization of the words being used.

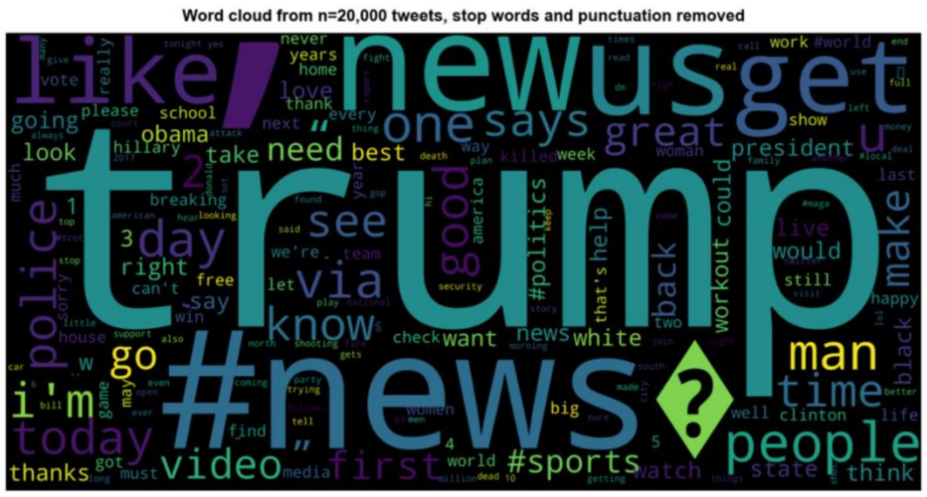


Figure 4 – Word cloud of a randomized sample of n=20,000 tweets, with stop words and punctuation removed.

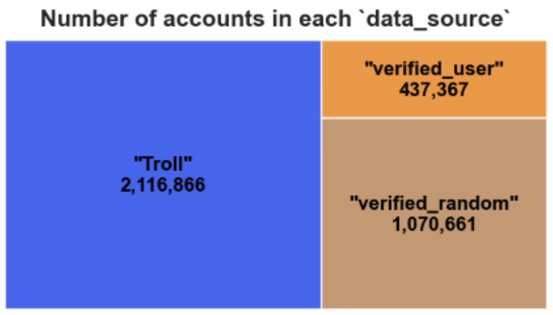


Figure 5 – Accounts per data source. Refer to Table 6, Table 7, and Table 8 in Appendix for details of "verified_" categories.

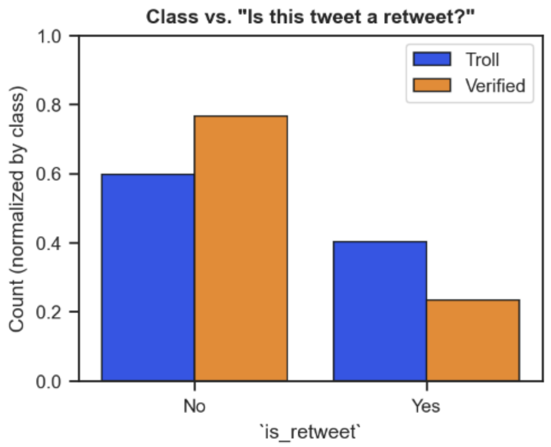


Figure 6 – A comparison of feature "is_retweet" separated by troll and verified classes.

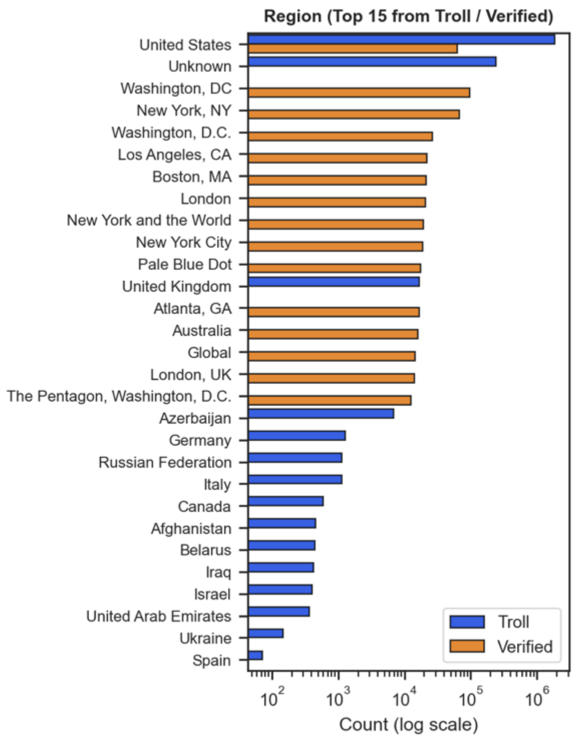


Figure 7 – A histogram showing frequency of the top 15 regions for both troll and verified tweets.

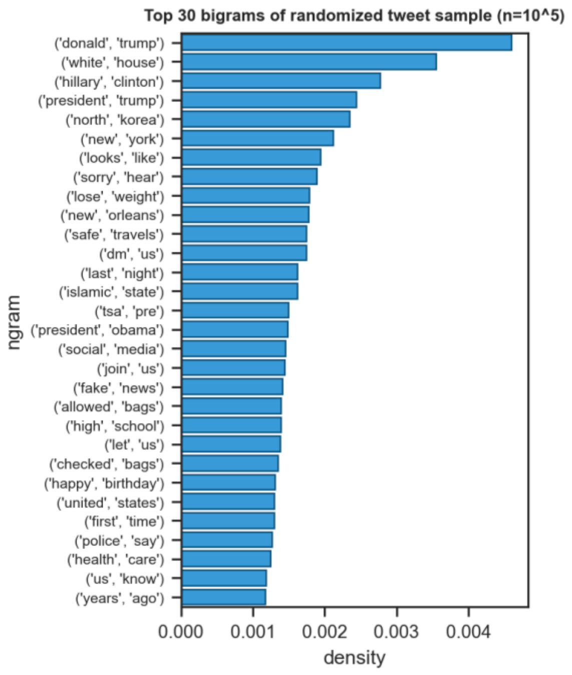


Figure 8 – Top 30 bigrams observed in a randomized sample of n=10^5 tweets, ranked by density.

3.4. Feature Engineering Results

3.4.1. Following Ratio

As the graph in **Figure 9** depicts, when compared against the account category, it becomes apparent that verified users tend to have the smallest median following ratio, meaning that they have far more followers than accounts they follow. Whereas some trolls tend to follow more accounts than they have followers. The closer to 1.0 indicates that for each account a user follows, hypothetically that account follows them back. This can represent the scope of influence an account has, while helping to normalize accounts with large numbers of followers and following.

3.4.2. Tweet Length

The graph in **Figure 10** shows the distribution of tweet length for the entire dataset among each of our two classes, using Pythons built-in `len()` function. There appears to be some difference in the range of (40, 110) characters where troll tweets tend to be more prevalent, thus indicating that Trolls may have a standard-length guideline they follow for tweets.

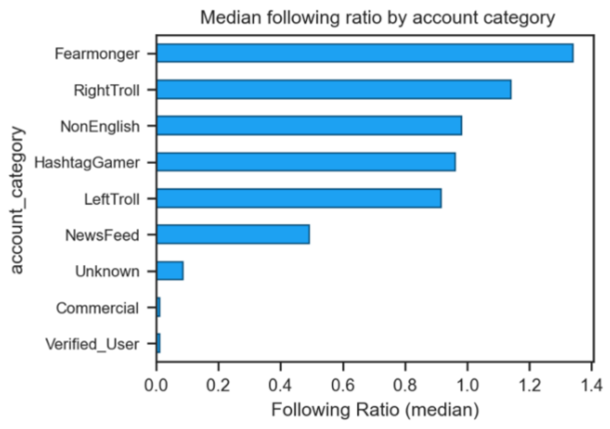


Figure 9 – Median following ratio by account category

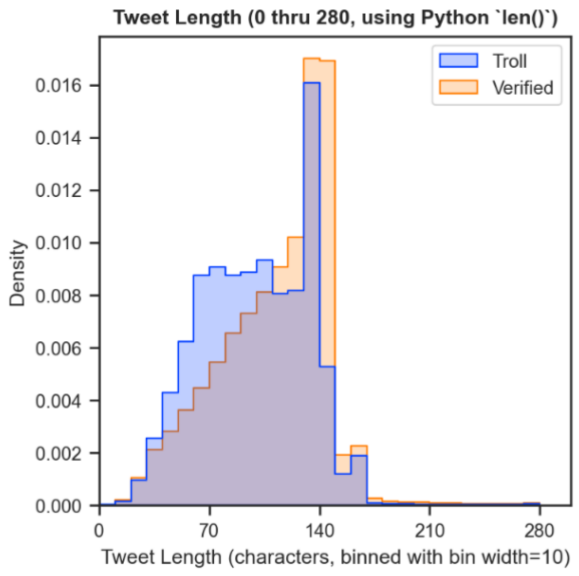


Figure 10 – Tweet length by class

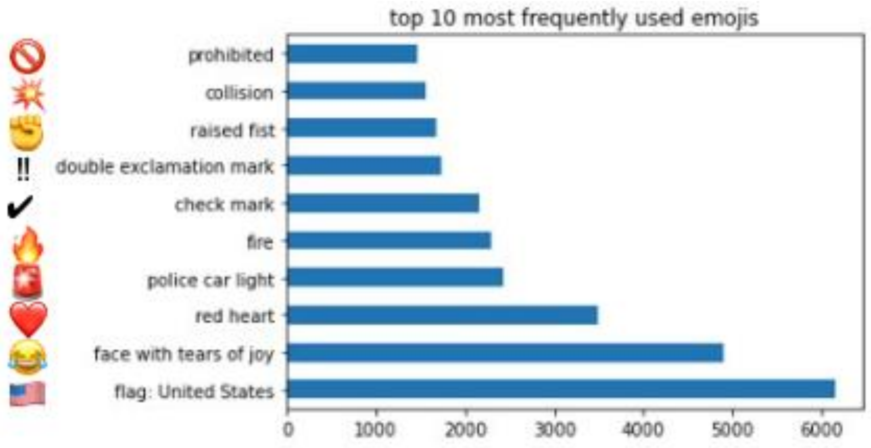


Figure 11 – Top 10 emojis (entire dataset)

3.4.3. Emoji Count

For verified users, the most frequently used emoji was "face with tears of joy", whereas for trolls it was "flag: United States". There are many overlapping emojis used by both groups, but it is evident that the American Flag played a larger role in the context of tweets made by trolls, which makes sense given the agenda of influencing American politics. **Figure 11** shows the top 10 emoji characters used across both dataset classes. Given that the overall prevalence of emojis was small but the variety was large, the 17 most frequently used emojis were encoded as binary features to determine if any single, specific emoji could be linked to the Troll tweets.

3.5. Modeling Results

3.5.1. Modeling Pipeline #1 Results

As described in section 2.5.1, each of our five selected BERT-based PLMs was fine-tuned using a subset of the tweets from our overall dataset. The size of tweet subset used for fine tuning ("fine-tune corpus size") was varied across a set of predetermined quantities of tweets: 5,000, 10,000, 25,000, 50,000, 100,000, and 200,000 tweets. The performance metrics calculated against evaluation datasets for each fine-tuned model using each fine-tune corpus size are reported in **Table 2** below.

3.5.2. Modeling Pipeline #2 Results

The three models chosen all performed well on the 3 feature subsets (20 account features, 29 tweet text features, 15 reduced dimensionality features) and entire domain of features. When tested against the tweet text only features (Notebook 7b), the accuracy scores were lowest around 0.75, but otherwise all other feature sets tested above 0.95 for accuracy. Results for Modeling Pipeline #2 are provided in **Table 3** below.

FEATURES OF IMPORTANCE

Utilizing the Logistic Regression model coefficients method, f-value and p-value scores, and correlation heat maps, we were able to deduce feature importance and used the results to reduce dimensionality. Of the original 49 features, 15 features were determined to have substantial coefficients and correlation scores. Features such as following ratio, multi authors, and unknown region had the highest positive coefficients, whereas features such as tweet length, sentiment, and numerous regions had the highest negative coefficients. Less impactful or redundant features such as following, followers, region, has URL, and all binary encoded emoji columns were removed to reduce dimensionality. None of the emoji features scored highly enough to substantiate our hypothesis. See **Figure 23**, **Figure 24**, **Figure 25**, and **Figure 26** in Appendix for correlation heat maps and f/p- value charts.

3.5.3. Modeling Pipeline #3 Results

Results for Modeling Pipeline #3 are provided in **Table 4**. Both multimodal models listed were trained using a sampling of 1 million tweets, with any tweets used for fine-tuning the Modeling Pipeline #1 transformer filtered out.

The two highest performing fine-tuned transformers were RoBERTa-200k and BERTweet-200k, with RoBERTa-200k narrowly taking the leading performance position. This pattern was repeated in the multimodal model results, with RoBERTa-200k achieving slightly higher metrics. With that said, both multimodal models were at near-perfect classification performance across the metrics measured so the minute differences in performance are likely insignificant.

What is significant, however, is the amount of training time required between the two multimodal models. When accounting for both time to fine-tune and time to train the multimodal model, RoBERTa-200k required 2.5x greater training time when compared to BERTweet-200k. This is a strong endorsement of the tweet-focused pre-training methods used for BERTweet and their applicability to our classification task.

Table 2 – Modeling Pipeline #1 – Performance Metric Results (Note 1)

Model	*Fine-Tune Corpus Size* (Note 2)	Accuracy	F1	Precision	Recall	ROC AUC	Brier Score (Note 3)
BERT	5k	0.856	0.859	0.844	0.874	0.856	0.144
	10k	0.860	0.856	0.882	0.832	0.860	0.140
	25k	0.877	0.875	0.888	0.862	0.877	0.123
	50k	0.892	0.892	0.898	0.885	0.892	0.108
	100k	0.901	0.900	0.903	0.897	0.901	0.100
	200k	0.912	0.912	0.916	0.907	0.912	0.088
DistilBERT	5k	0.846	0.844	0.856	0.832	0.846	0.154
	10k	0.860	0.860	0.860	0.859	0.860	0.140
	25k	0.874	0.874	0.879	0.869	0.874	0.126
	50k	0.878	0.878	0.877	0.878	0.878	0.123
	100k	0.897	0.896	0.904	0.889	0.897	0.103
	200k	0.909	0.909	0.913	0.903	0.909	0.091
RoBERTa	5k	0.871	0.869	0.882	0.856	0.871	0.129
	10k	0.892	0.891	0.899	0.883	0.892	0.108
	25k	0.904	0.903	0.910	0.896	0.904	0.096
	50k	0.913	0.912	0.916	0.909	0.913	0.088
	100k	0.916	0.915	0.924	0.906	0.916	0.084
	200k	**{0.925}**	**{0.924}**	**{0.932}**	0.917	**{0.925}**	**{0.075}**
BERTweet	5k	0.890	0.893	0.867	**{0.922}**	0.890	0.110
	10k	0.890	0.890	0.888	0.891	0.890	0.111
	25k	0.906	0.907	0.903	0.910	0.906	0.094
	50k	0.913	0.912	0.919	0.905	0.913	0.087
	100k	0.919	0.919	0.926	0.910	0.919	0.081
	200k	0.924	0.923	0.928	0.919	0.924	0.076
TwHIN-BERT	5k	0.883	0.883	0.882	0.884	0.883	0.117
	10k	0.890	0.889	0.897	0.881	0.890	0.110
	25k	0.905	0.904	0.912	0.896	0.905	0.095
	50k	0.904	0.904	0.904	0.904	0.904	0.096
	100k	0.907	0.906	0.913	0.899	0.907	0.093
	200k	0.917	0.916	0.919	0.914	0.917	0.083

Table Notes:

(1) Within a given metric column, more densely shaded cells represent higher performance for that metric. Color scales are independent between metric columns. Highest performance values in each column are denoted with **{bold text surrounded by curly braces}**.

(2) Fine-tune Corpus Size – Values are given in thousands of tweets and represent the quantity of tweets used in training dataset. Additional, non-overlapping tweets were used for evaluation such that a 5-to-1 quantity ratio of train-to-evaluation dataset size was imposed. For example, for a corpus size of “50k”, the training set contained 50,000 tweets and the evaluation set contained 10,000 tweets, with no overlap between the sets.

(3) Brier Score – Lower values indicate higher performance. For all remaining metrics, higher values indicate higher performance.

Table 3 – Modeling Pipeline #2 – Performance Metric Results

Model	Number of Features	Accuracy	Precision	Recall	ROC AUC
Logistic Regression	15	0.956	0.951	0.961	0.987
	49	0.993	0.994	0.979	0.991
Decision Tree	15	0.99	0.987	0.992	0.997
	49	0.998	0.999	0.997	0.999
XGBoost	15	0.996	0.997	0.995	0.996
	49	**0.999**	**0.999**	**0.999**	**0.999**

Table 4 – Modeling Pipeline #3 – Performance Metric Results

Multimodal Model / Notes	**Accuracy**	**F1**	**Precision**	**Recall**	**ROC AUC**	**Brier Score** (Note 1)
BERTweet • Fine-tuned Model: `BERTweet-200k` • Training Time ○ Fine-Tuning: 5 hr 46 min ○ Multimodal: 4 hr 8 min ○ Total: 9 hr 54 min	0.998	0.998	0.998	0.999	0.999	0.001
RoBERTa • Fine-tuned Model: `RoBERTa-200k` • Training Time ○ Fine-Tuning: 15 hr 10 min ○ Multimodal: 9 hr 40 min ○ Total: 24 hr 50 min	0.999	0.999	0.999	0.999	0.999	0.001

Table Notes:
(1) Brier Score – Lower values indicate higher performance.

4. Discussion

4.1. Validity of Twitter Verification Criteria during Dataset Timeframe

As noted in section 1.4.2, Twitter's policy for account verification has recently changed. As has been demonstrated, a foundational assumption of our study is that a tweet obtained from a verified account is authentic and able to be labeled as such. This begs the question of whether this assumption is still valid.

Our assessment on this assumption is this: even though portions of our data were obtained recently (immediately prior to the policy change), the period during which our data was generated does not overlap with the revised policy. We can verify (using archived versions of the verification policy pages from Twitter's website) that the policy was consistent with how we have presented it both during the time period we acquired data (Sept/Oct 2022) [50] and during the time-period we are studying (years 2013-2017) [8].

Based on this, we conclude our original assumption is valid. Further, despite the uncertain future of the Twitter verification program or its policies, we believe there are still valuable insights to be gained by undertaking this study on the acquired and pre-processed data we have herein described.

4.2. Comparable Results

Given our high accuracy scores and overall performance metrics, we undertook a literature review to validate that our models performed within established ranges. Researchers at Duzce University in Turkey tested BERT, ELMo, and GloVe on a similar but much smaller Troll dataset (expanded from the original work of Zannetou et. al 2018), with overlapping time frames and related features [21,51,52]. Their work produced accuracy scores above 84% for BERT alone and an AUC of 91.5% for BERT combined with a Convolutional Neural Network (CNN).

Work done by Chua Chin Hon, using the same base Troll dataset and fine-tuned DistilBERT on 90k tweets, achieved accuracy of 92% on the validation dataset [53]. In comparison to Modeling Pipeline #2 results of over 99% accuracy, work done by Maiia Bakhova on the same Troll dataset showed equally high performance using similar features and a CatBoost Classifier [32].

5. Conclusions

Tackling a problem as complex as disinformation is no small task. We set out to accomplish three goals within the scope of this project in order to build upon existing research in the realm of NLP, transformer models, transfer learning, and binary classification tasks as they relate to the Linvell & Warren Russian Troll dataset. Our results were noteworthy in the sense that the multimodal model, to the best of our knowledge, has not been utilized on this dataset, and the results are promising.

Given the promising results of the multimodal model, we envision this having potential applications in numerous arenas such as: the consumer realm (fake review detection based on text, ratings, and account features); medical (biometric data combined with medical chart notes); and national security (social media monitoring and other intelligence gathering data feeds).

We also demonstrated that traditional ML models have both time and resource advantages over transformer models and deep learning in this context, but the former require much more data-preprocessing and feature engineering to yield comparable results to the latter. Using only the entry-level NLP we performed on the tweet context alone, the traditional ML models did not exceed 75% accuracy (Notebook 7b), whereas the worst performing fine-tuned BERT model scored over 85% for accuracy. A key takeaway from the traditional ML models was that the account features provide reliable indicators of disinformation that can readily be monitored and screened to provide stronger security controls.

5.1. Future Work

Although our results are very promising, there are many avenues for future work that can be pursued. One such example is to engineer other features included in the literature that incorporate network features, stemming from the graph-theoretical approach to social network theory. A book section on "Feature Engineering for Social Bot Detection" from the 2018 "Feature Engineering for Machine Learning" includes features generated via "interaction and hashtag co-occurrence networks," as well as content and language-specific features like word counts (which we include), but also expand to text entropy as well as frequency and proportion of POS tags in a given Tweet using "verbs, nouns, adjectives, modal auxiliaries, pre-determiners, interjections, adverbs, wh-, and pronouns" [54]. According to this same book section, the authors find that the top five features according to Random Forest include the number of friends, number of favorites, mentioned friends' mean Tweet count, user's number of followers, and account age [54].

One interesting network feature (that was the seventh top feature) was the mention network mean edge strength, in which a high mean edge strength can indicate a tightly connected network with frequent, intense interactions between users, with a low mean edge strength suggesting a looser network with weaker connections. This metric can also identify influential users who have strong connections with many other users in the mention network, such as an influencer or influential troll account within that given mention network. However, this is not the sole indicator of identifying an influencer, as the other features must also be considered [54].

Another interesting avenue would be to explore how well our models generalize on other more recent datasets or even incorporate multi-lingual tweets. Given that this dataset explored a distinct disinformation campaign during a presidential election, the nuances and subtleties used by the IRA in this campaign may not translate to other time periods or generalize well. However, lack of publicly available labeled datasets and changes to Twitter's verification policy complicate this future effort. The temporal aspect is another exciting prospect that we did not delve into, but likely would yield beneficial results given the frequency we observed Troll tweets in comparison to average users.

Lastly, exploring BERT explainability methods and using the package `transformers-interpret` could yield some insight into what features of the text BERT identified as important and alleviate concerns over biased AI and ethical issues [55,56]. This could provide more substantial discussion of what text features are identifiers of disinformation and enhance the understanding of various BERT models.

6. Acknowledgements

The authors would like to acknowledge and thank Dr. Milad Toutounchian (Drexel University) for guidance, encouragement, and feedback provided during the course of this capstone project, especially relating to the application of data science and machine learning concepts. The authors would also like to acknowledge and thank Dr. Asta Zelenkauskaite (Drexel University) for providing valuable insights on coordinated, online disinformation campaigns as well as the academic research landscape surrounding this topic. This project would not have been possible without the work of the troll dataset's original researchers (D. Linvill and P. Warren of Clemson University) as well as FiveThirtyEight's Oliver Roeder. The availability of Twitter data through Twitter's API and the no-cost incarnation of its Academic Research access level allowed the troll dataset to be balanced with data from verified Twitter users. Finally, the many open-source software packages used by this project, either directly or indirectly, enabled not only for an exciting analysis to be performed but also for the mechanics of that analysis to be examined and understood.

7. References

[1] A. Zelenkauskaite, Creating Chaos Online, University of Michigan Press, Ann Arbor, MI, 2022. https://doi.org/10.3998/mpub.12237294.

[2] Office of the Director of National Intelligence, Assessing Russian Activities and Intentions in Recent US Elections, 2017. https://www.dni.gov/files/documents/ICA_2017_01.pdf (accessed February 26, 2023).

[3] K.H. Jamieson, Cyber-war: How Russian hackers and trolls helped elect a president: What We Don't, Can't, and Do Know, Oxford University Press, 2020.

[4] Office of the Director of National Intelligence, Assessing Russian Activities and Intentions in Recent US Elections, (2017). https://www.dni.gov/files/documents/ICA_2017_01.pdf (accessed February 26, 2023).

[5] Twitter Public Policy, Update on Twitter's review of the 2016 US election, 2018. https://blog.twitter.com/official/en_us/topics/company/2018/2016-election-update.html (accessed February 26, 2023).

[6] O. Roeder, fivethirtyeight/russian-troll-tweets, GitHub. (2018). https://github.com/fivethirtyeight/russian-troll-tweets/releases/tag/v2.0 (accessed February 24, 2023).

[7] H. Tsukayama, Twitter is officially doubling the character limit to 280 - The Washington Post, Washington Post. (2017). https://www.washingtonpost.com/news/the-switch/wp/2017/11/07/twitter-is-officially-doubling-the-character-limit-to-280/ (accessed February 26, 2023).

[8] Twitter, About verified accounts (Web Archived Version, 2017), Twitter Help Center. (2017). https://web.archive.org/web/20171231002426/https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts (accessed December 30, 2017).

[9] Twitter, Legacy Verification policy, Twitter Help Center. (2022). https://web.archive.org/web/20221109153246/https://help.twitter.com/en/managing-your-account/legacy-verification-policy (accessed November 8, 2022).

[10] What is transfer learning?, TensorFlow Tutorials. (2022). https://www.tensorflow.org/js/tutorials/transfer/what_is_transfer_learning (accessed March 18, 2023).

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention Is All You Need, (2017). http://arxiv.org/abs/1706.03762.

[12] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, X. Huang, Pre-trained Models for Natural Language Processing: A Survey, Sci China Technol Sci. 63 (2020) 1872–1897. https://doi.org/10.1007/s11431-020-1647-3.

[13] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, (2018). http://arxiv.org/abs/1810.04805.

[14] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving Language Understanding by Generative Pre-Training, (2018). https://gluebenchmark.com/leaderboard (accessed March 18, 2023).

[15] U. Ankit, Transformer Neural Networks: A Step-by-Step Breakdown, Built In. (2022). https://builtin.com/artificial-intelligence/transformer-neural-network (accessed March 18, 2023).

[16] Explanation of BERT Model - NLP, GeeksforGeeks. (2022). https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/ (accessed March 18, 2023).

[17] C. Horan, 10 Things You Need to Know About BERT and the Transformer Architecture That Are Reshaping the AI Landscape, Neptune.Ai. (2023). https://neptune.ai/blog/bert-and-the-transformer-architecture (accessed March 18, 2023).

[18] A. Bessi, E. Ferrara, Social bots distort the 2016 U.S. Presidential election online discussion, First Monday. 21 (2016). https://doi.org/10.5210/FM.V21I11.7090.

[19] S. Zannettou, T. Caulfield, W. Setzer, M. Sirivianos, G. Stringhini, J. Blackburn, Who let the trolls out? towards understanding state-sponsored trolls, WebSci 2019 - Proceedings of the 11th ACM Conference on Web Science. 10 (2019) 353–362. https://doi.org/10.1145/3292522.3326016.

[20] A. Zelenkauskaite, P. Toivanen, J. Huhtamäki, K. Valaskivi, Shades of hatred online: 4chan duplicate circulation surge during hybrid media events, First Monday. (2021). https://doi.org/10.5210/FM.V26I1.11075.

[21] S. Zannettou, T. Caulfield, E. De Cristofaro, M. Sirivianos, G. Stringhini, J. Blackburn, Disinformation Warfare: Understanding State-Sponsored Trolls on Twitter and Their Influence on the Web, (2018).

[22] S. Zannettou, T. Caulfield, B. Bradlyn, E. De Cristofaro, G. Stringhini, J. Blackburn, Characterizing the Use of Images in State-Sponsored Information Warfare Operations by Russian Trolls on Twitter, Proceedings of the International AAAI Conference on Web and Social Media. 14 (2020) 774–785. https://doi.org/10.1609/ICWSM.V14I1.7342.

[23] E. Dresner, S.C. Herring, Functions of the Nonverbal in CMC: Emoticons and Illocutionary Force, Communication Theory. 20 (2010) 249–268. https://doi.org/10.1111/J.1468-2885.2010.01362.X.

[24] A. Konrad, S.C. Herring, D. Choi, Sticker and Emoji Use in Facebook Messenger: Implications for Graphicon Change, Journal of Computer-Mediated Communication. 25 (2020) 217–235. https://doi.org/10.1093/JCMC/ZMAA003.

[25] FiveThirtyEight, Russian Troll Tweets, Github. (2018). https://github.com/fivethirtyeight/russian-troll-tweets/.

[26] D.L. Linvill, P.L. Warren, Troll Factories: Manufacturing Specialized Disinformation on Twitter, Polit Commun. 37 (2020) 447–467. https://doi.org/10.1080/10584609.2020.1718257.

[27] Twitter, Tweet Downloader, Twitter API Tools. (n.d.). http://web.archive.org/web/20221119170903/https://developer.twitter.com/apitools/downloader (accessed February 26, 2023).

[28] Twitter Developer Platform, Search Tweets - How to build a query - Operator types, Twitter API v2 Docs. (n.d.). https://developer.twitter.com/en/docs/twitter-api/tweets/search/integrate/build-a-query#types (accessed March 18, 2023).

[29] Stop word - Wikipedia, (n.d.). https://en.wikipedia.org/wiki/Stop_word (accessed March 18, 2023).

[30] The Unicode Consortium, Unicode Index of Public Emoji 14.0, (n.d.). https://unicode.org/Public/emoji/14.0/.

[31] Salesforce, Region classification for sources and posts in Social Studio, Salesforce.Com Help. (n.d.).

[32] M. Bahkova, Russian Troll Detection By Their Tweets, DataScienceCentral. (2022). https://www.datasciencecentral.com/russian-troll-detection-by-their-tweets/ (accessed March 4, 2023).

[33] JRX Toronto, Unicode Cyrillic, (n.d.). https://jrgraphix.net/r/Unicode/0400-04FF (accessed March 15, 2023).

[34] Robyn Speer, ftfy (Version 5.5), Zenodo. (2019). https://doi.org/10.5281/zenodo.2591652.

[35] C.J. Hutto, E.E. Gilbert, VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text, Eighth International Conference on Weblogs and Social Media (ICWSM-14). (2014).

[36] V. Sanh, L. Debut, J. Chaumond, T. Wolf, DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, (2019). https://doi.org/10.48550/arxiv.1910.01108.

[37] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, P.G. Allen, RoBERTa: A Robustly Optimized BERT Pretraining Approach, (2019). https://doi.org/10.48550/arxiv.1907.11692.

[38] D. Quoc Nguyen, T. Vu, A. Tuan Nguyen, V. Research, BERTweet: A pre-trained language model for English Tweets, (2020) 9–14. https://doi.org/10.18653/V1/2020.EMNLP-DEMOS.2.

[39] X. Zhang, Y. Malkov, O. Florez, S. Park, B. McWilliams, J. Han, A. El-Kishky, TwHIN-BERT: A Socially-Enriched Pre-trained Language Model for Multilingual Tweet Representations, (2022). https://doi.org/10.48550/arxiv.2209.07562.

[40] XGBoost Developers, XGBoost Documentation, Revision 36ad1605. (n.d.). https://xgboost.readthedocs.io/en/stable/ (accessed March 17, 2023).

[41] D.J. Toth, Binary Classification: XGBoost Hyperparameter Tuning Scenarios by Non-exhaustive Grid Search and Cross-Validation, Towards Data Science. (2021). https://towardsdatascience.com/binary-classification-xgboost-hyperparameter-tuning-scenarios-by-non-exhaustive-grid-search-and-c261f4ce098d (accessed March 17, 2023).

[42] K. Gu, A. Budhkar, A Package for Learning on Tabular and Text Data with Transformers, in: Proceedings of the Third Workshop on Multimodal Artificial Intelligence, Association for Computational Linguistics, Mexico City, Mexico, 2021: pp. 69–73.

[43] Ken Gu, How to Incorporate Tabular Data with HuggingFace Transformers, Medium. (2020). https://medium.com/georgian-impact-blog/how-to-incorporate-tabular-data-with-huggingface-transformers-b70ac45fcfb4 (accessed March 18, 2023).

[44] K. Gu, A. Budhkar, A Package for Learning on Tabular and Text Data with Transformers, Multimodal Artificial Intelligence, MAI Workshop 2021 - Proceedings of the 3rd Workshop. (2021) 69–73. https://doi.org/10.18653/V1/2021.MAIWORKSHOP-1.10.

[45] Brier score - Wikipedia, (n.d.). https://en.wikipedia.org/wiki/Brier_score (accessed March 17, 2023).

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research. 12 (2012) 2825–2830. https://arxiv.org/abs/1201.0490v4 (accessed March 17, 2023).

[47] A. Bhandari, Guide to AUC ROC Curve in Machine Learning : What Is Specificity?, Analytics Vidhya. (2020). https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/ (accessed March 18, 2023).

[48] D.L. Linvill, P.L. Warren, Troll Factories: Manufacturing Specialized Disinformation on Twitter, Polit Commun. 37 (2020) 447–467. https://doi.org/10.1080/10584609.2020.1718257.

[49] Wikipedia, Pale Blue Dot, Wikipedia. (n.d.). https://en.wikipedia.org/wiki/Pale_Blue_Dot (accessed December 7, 2022).

[50] Twitter, About Verified Accounts (Web Archived Version, 2022), Twitter Help Center. (2022). https://web.archive.org/web/20220901000754/https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts (accessed August 31, 2022).

[51] S. Yilmaz, S. Zavrak, Troll Tweet Detection Using Contextualized Word Representations, n.d. https://arxiv.org/pdf/2207.08230.pdf.

[52] L. Miao, M. Last, M. Litvak, Detecting Troll Tweets in a Bilingual Corpus, in: Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), European Language Resources Association (ELRA), Marseille, 2020: pp. 6247–6254.

[53] Chua Chin Hon, Using Data Science To Uncover State-Backed Trolls On Twitter, Towards Data Science. (2019). https://towardsdatascience.com/using-data-science-to-uncover-state-backed-trolls-on-twitter-dc04dc749d69 (accessed March 4, 2023).

[54] O. Varol, C.A. Davis, F. Menczer, A. Flammini, Feature Engineering for Social Bot Detection, in: G. Dong, H. Liu (Eds.), Feature Engineering for Machine Learning and Data Analytics, CRC Press, 2018: pp. 311–334. https://books.google.com/books?id=661SDwAAQBAJ.

[55] M. Szczepański, M. Pawlicki, R. Kozik, M. Choraś, New explainability method for BERT-based model in fake news detection, Sci Rep. 11 (2021) 23705. https://doi.org/10.1038/s41598-021-03100-6.

[56] S.N. Gupta, BERT Explainability, Medium. (2021). https://medium.com/aiguys/bert-explainability-5b54cff01407 (accessed March 18, 2023).

Appendix

Additional Figures

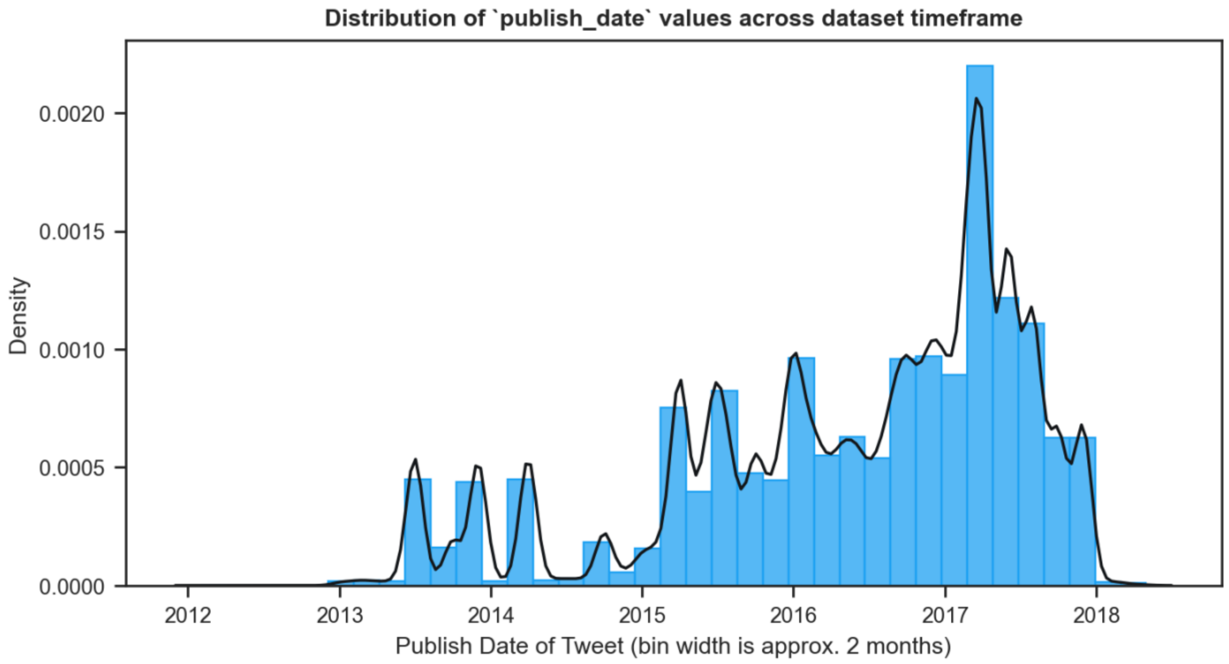


Figure 12 – Distribution of publish date (bin width = ~2 month) observed in a randomized sample of n=20,000 tweets.

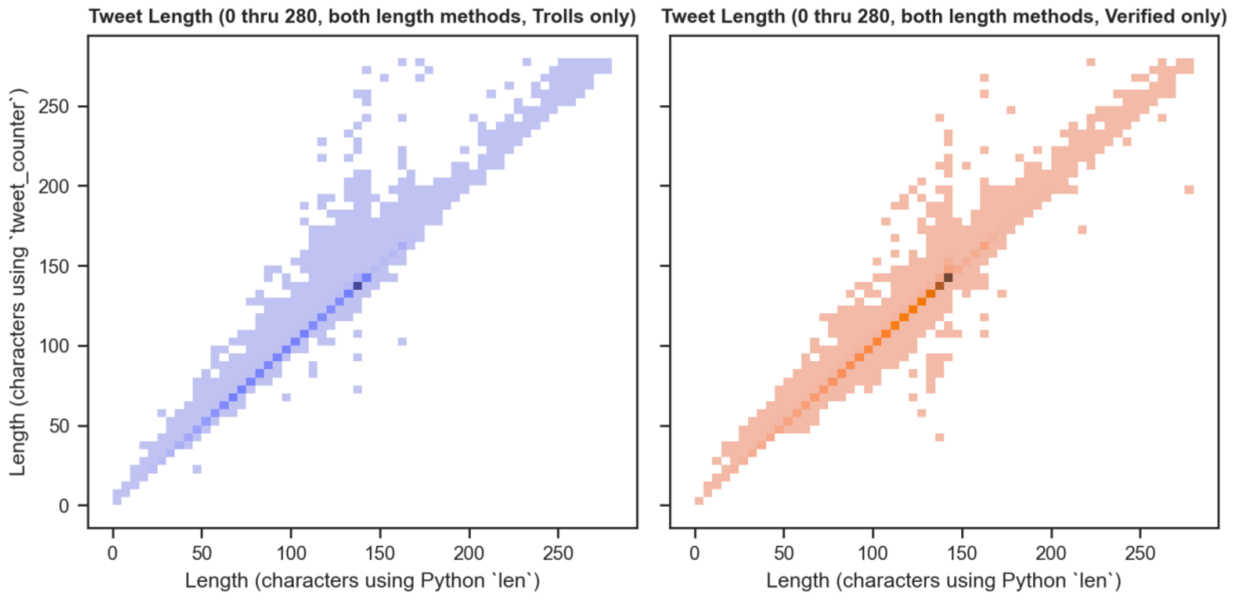


Figure 13 – Comparison of distributions of tweet length when calculated by either the Python `len()` function or an approximation of Twitter's distinct character counting method. The distribution for "Verified only" tweets appears to stay closer to the diagonal (indicating the two methods generally agree more). By contrast, the "Trolls only" distribution appears to skew upward from the diagonal, suggesting the Twitter method considers tweets to be longer than their simple Python len character count.

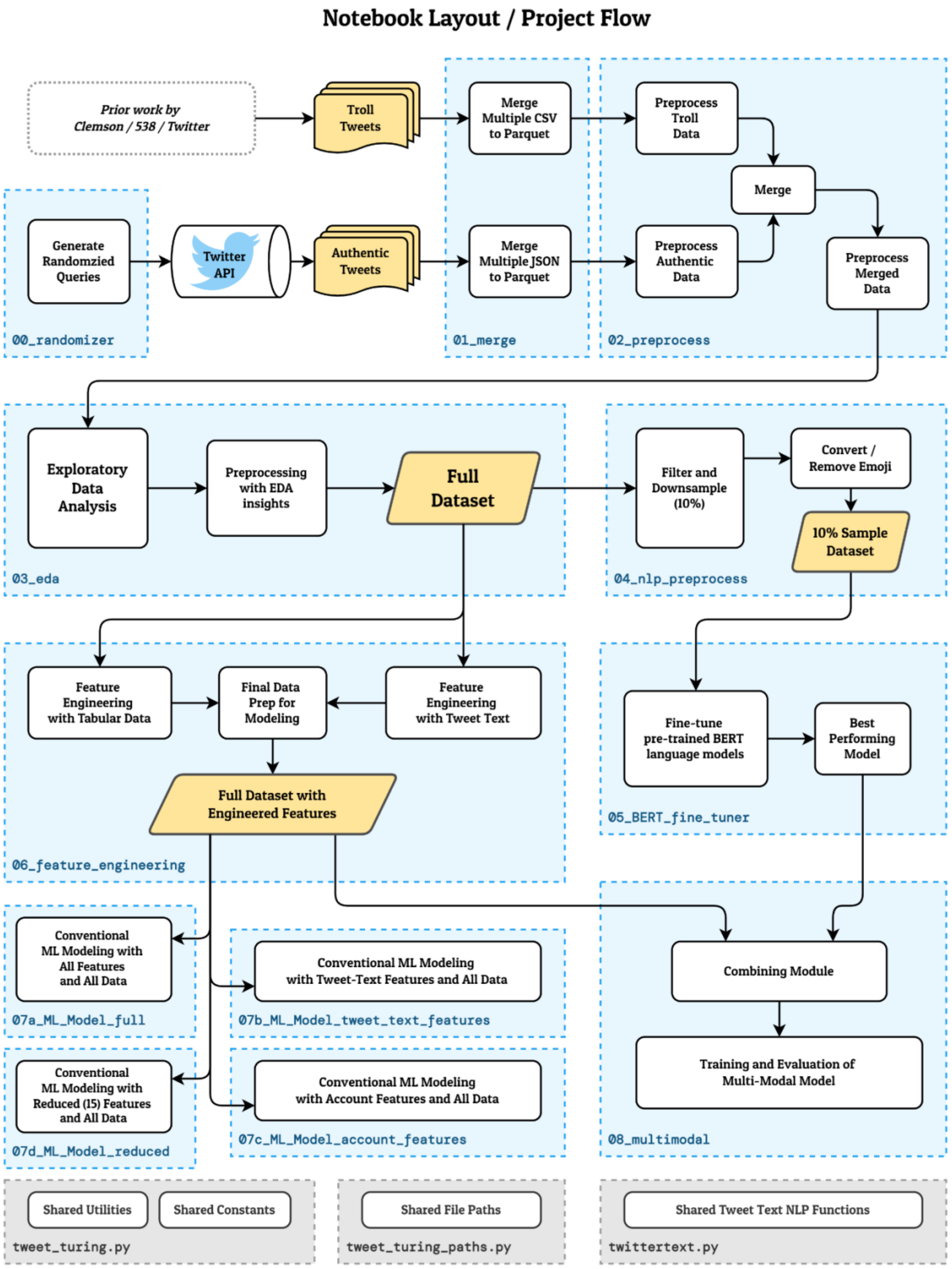

Notebook Layout / Project Flow
Prior work by Clemson / 538 / Twitter
Troll Tweets
Merge Multiple CSV to Parquet
Preprocess Troll Data
Merge
Generate Randomzied Queries
Twitter API
Authentic Tweets
Merge Multiple JSON to Parquet
Preprocess Authentic Data
Preprocess Merged Data
00_randomizer
01_merge
02_preprocess
Exploratory Data Analysis
Preprocessing with EDA insights
Full Dataset
03_eda
Filter and Downsample (10%)
Convert / Remove Emoji
10% Sample Dataset
04_nlp_preprocess
Feature Engineering with Tabular Data
Final Data Prep for Modeling
Feature Engineering with Tweet Text
Full Dataset with Engineered Features
06_feature_engineering
Fine-tune pre-trained BERT language models
Best Performing Model
05_BERT_fine_tuner
Conventional ML Modeling with All Features and All Data
07a_ML_Model_full
Conventional ML Modeling with Tweet-Text Features and All Data
07b_ML_Model_tweet_text_features
Combining Module
Training and Evaluation of Multi-Modal Model
Conventional ML Modeling with Reduced (15) Features and All Data
07d_ML_Model_reduced
Conventional ML Modeling with Account Features and All Data
07c_ML_Model_account_features
08_multimodal
Shared Utilities
Shared Constants
tweet_turing.py
Shared File Paths
tweet_turing_paths.py
Shared Tweet Text NLP Functions
twittertext.py

Figure 14 – A diagram summarizing the flow of our project and indicating which Jupyter notebook performs each function.

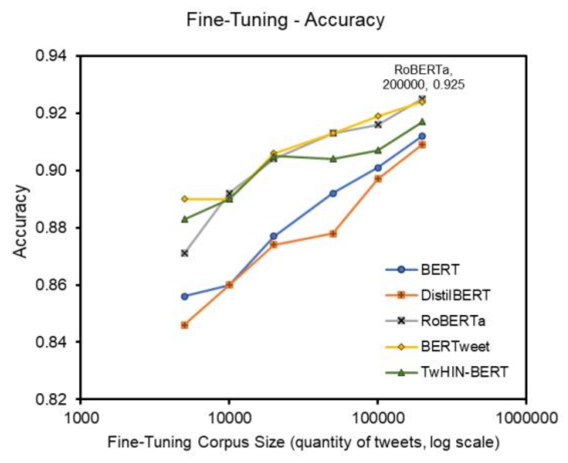


Figure 15 – Fine-tuned model accuracy with respect to fine-tuning corpus size

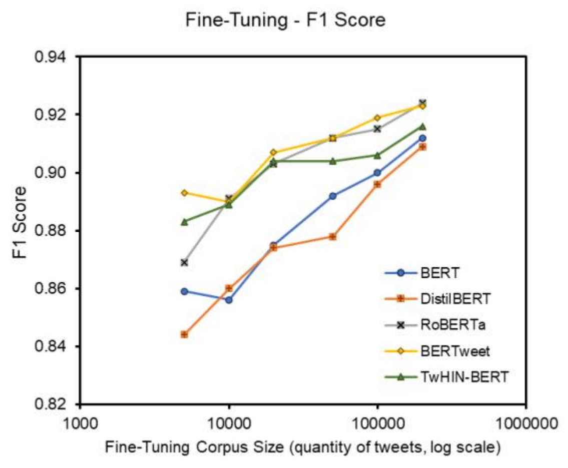


Figure 16 – Fine-tuned model F1 score with respect to fine-tuning corpus size

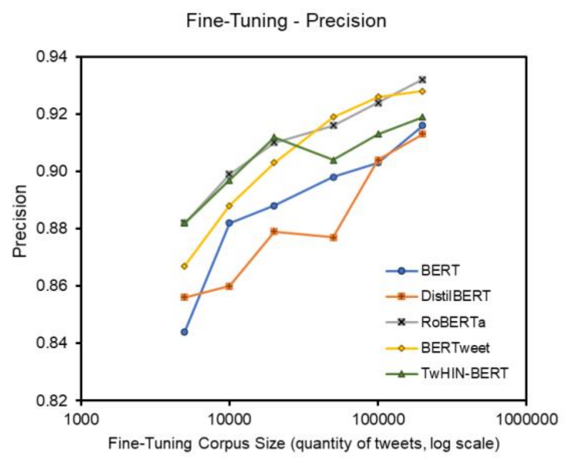


Figure 17 – Fine-tuned model precision with respect to fine-tuning corpus size

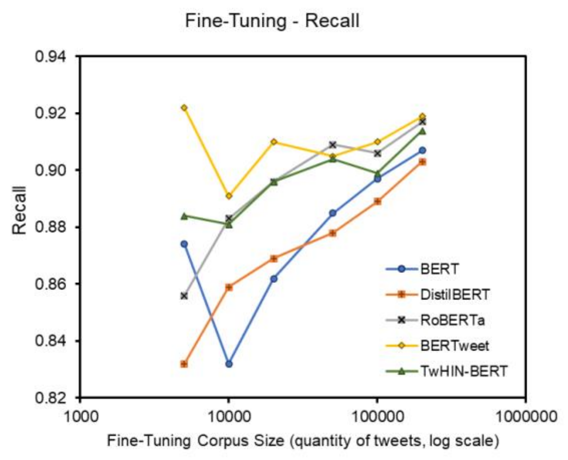


Figure 18 – Fine-tuned model recall with respect to fine-tuning corpus size

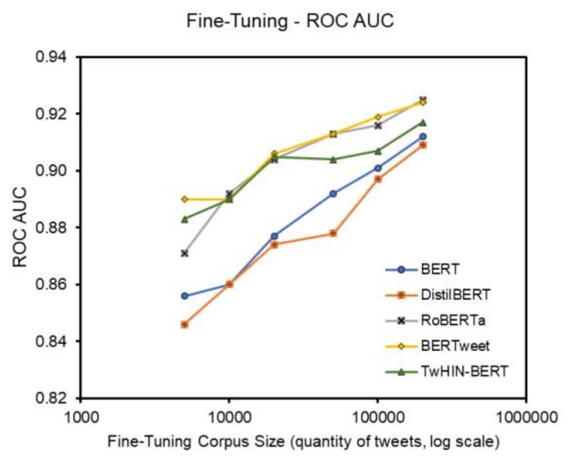


Figure 19 – Fine-tuned model ROC AUC with respect to fine-tuning corpus size

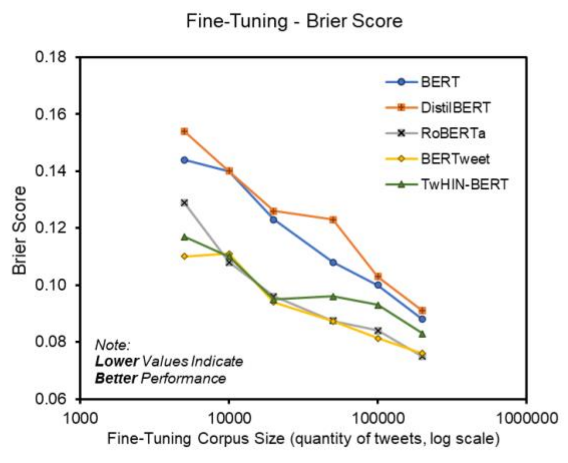


Figure 20 – Fine-tuned model Brier score with respect to fine-tuning corpus size

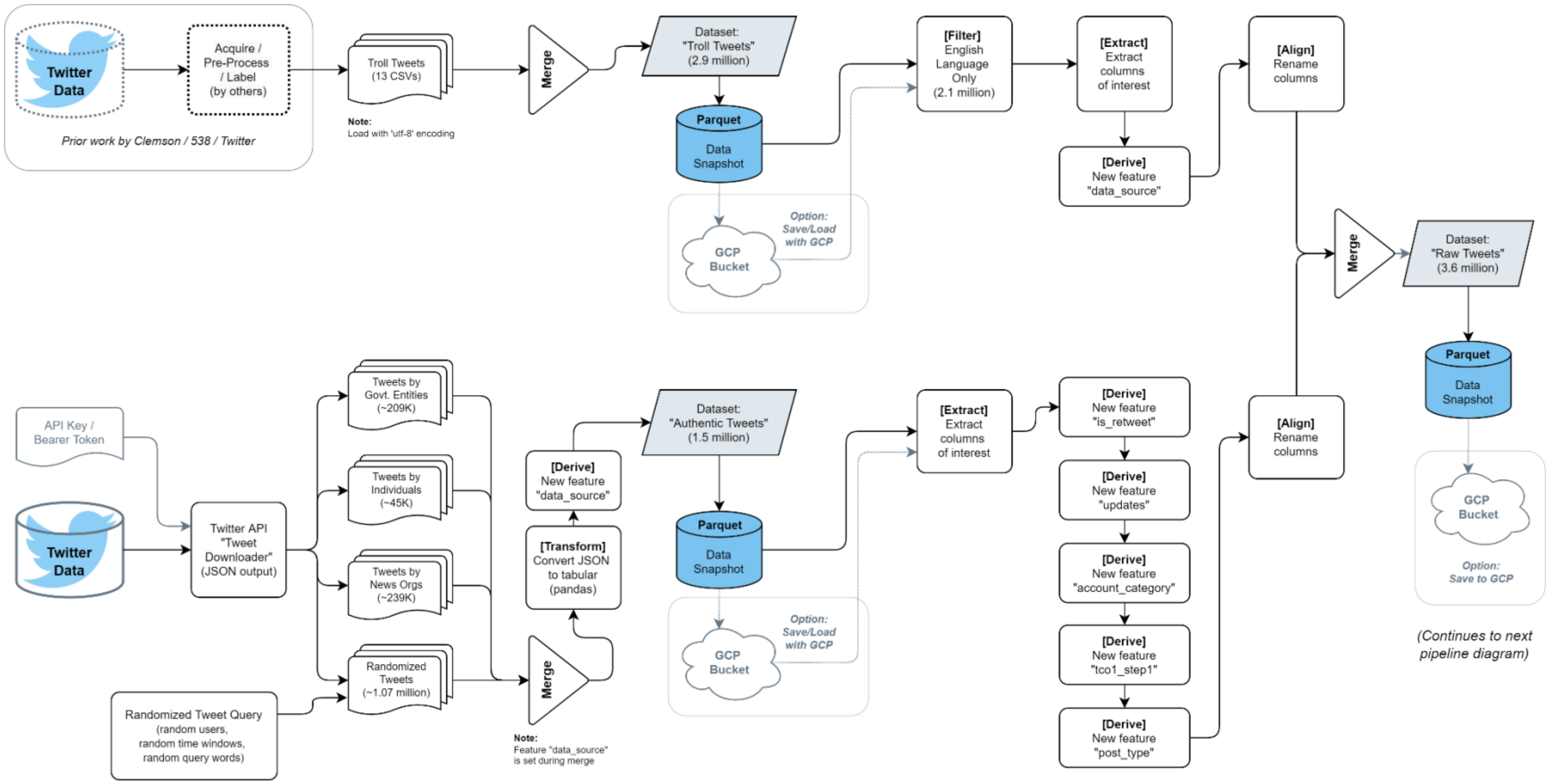

Twitter Data
Acquire / Pre-Process / Label (by others)
Prior work by Clemson / 538 / Twitter
Troll Tweets (13 CSVs)
Note: Load with 'utf-8' encoding
Merge
Dataset: "Troll Tweets" (2.9 million)
Parquet
Data Snapshot
GCP Bucket
Option: Save/Load with GCP
[Filter] English Language Only (2.1 million)
[Extract] Extract columns of interest
[Derive] New feature "data_source"
[Align] Rename columns
Merge
Dataset: "Raw Tweets" (3.6 million)
Parquet
Data Snapshot
GCP Bucket
Option: Save to GCP
(Continues to next pipeline diagram)
API Key / Bearer Token
Twitter Data
Twitter API "Tweet Downloader" (JSON output)
Tweets by Govt. Entities (~209K)
Tweets by Individuals (~45K)
Tweets by News Orgs (~239K)
Randomized Tweets (~1.07 million)
Randomized Tweet Query (random users, random time windows, random query words)
Merge
Note: Feature "data_source" is set during merge
[Transform] Convert JSON to tabular (pandas)
[Derive] New feature "data_source"
Dataset: "Authentic Tweets" (1.5 million)
Parquet
Data Snapshot
GCP Bucket
Option: Save/Load with GCP
[Extract] Extract columns of interest
[Derive] New feature "is_retweet"
[Derive] New feature "updates"
[Derive] New feature "account_category"
[Derive] New feature "tco1_step1"
[Derive] New feature "post_type"
[Align] Rename columns

Figure 21 – Data Acquisition, Preprocessing, and EDA Pipeline (1 of 2)

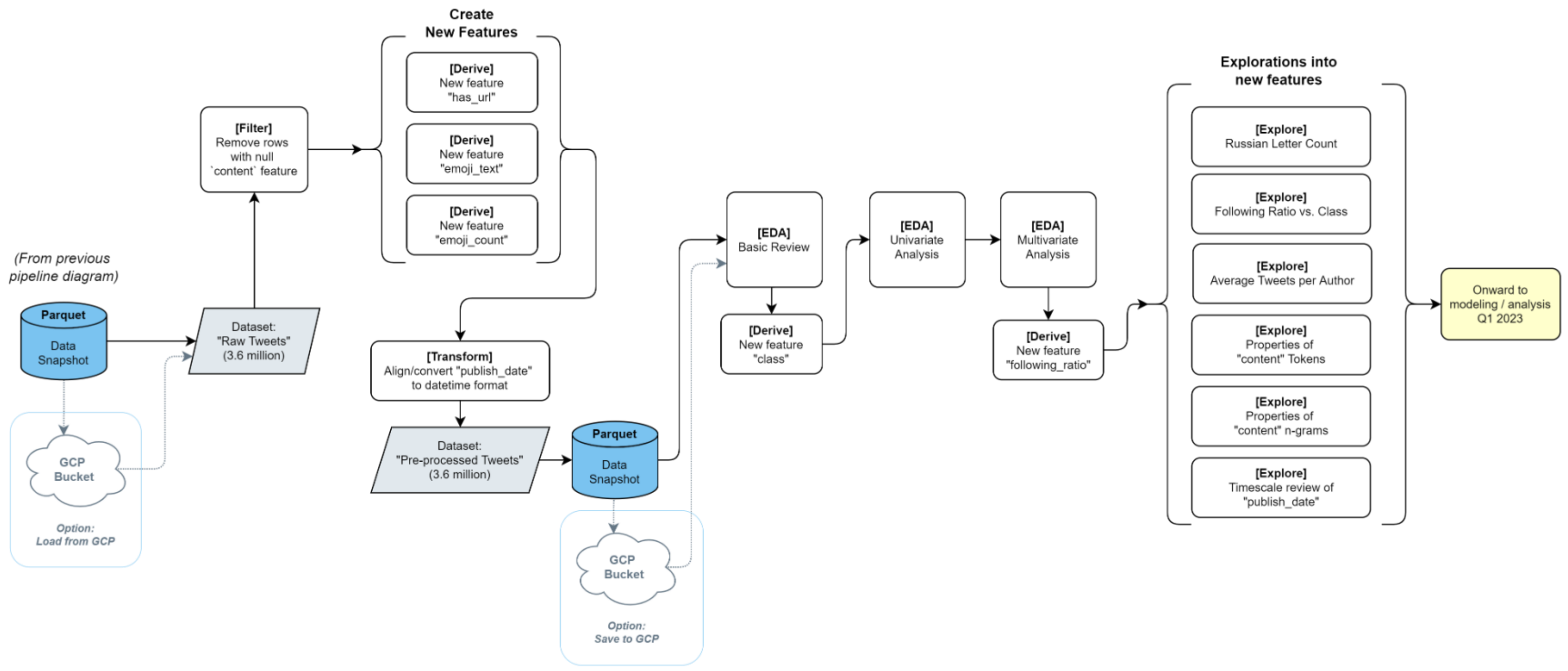

(From previous pipeline diagram)
Parquet
Data Snapshot
GCP Bucket
Option: Load from GCP
Dataset: "Raw Tweets" (3.6 million)
[Filter] Remove rows with null `content` feature
Create New Features
[Derive] New feature "has_url"
[Derive] New feature "emoji_text"
[Derive] New feature "emoji_count"
[Transform] Align/convert "publish_date" to datetime format
Dataset: "Pre-processed Tweets" (3.6 million)
Parquet
Data Snapshot
GCP Bucket
Option: Save to GCP
[EDA] Basic Review
[Derive] New feature "class"
[EDA] Univariate Analysis
[EDA] Multivariate Analysis
[Derive] New feature "following_ratio"
Explorations into new features
[Explore] Russian Letter Count
[Explore] Following Ratio vs. Class
[Explore] Average Tweets per Author
[Explore] Properties of "content" Tokens
[Explore] Properties of "content" n-grams
[Explore] Timescale review of "publish_date"
Onward to modeling / analysis Q1 2023

Figure 22 – Data Acquisition, Preprocessing, and EDA Pipeline (2 of 2)

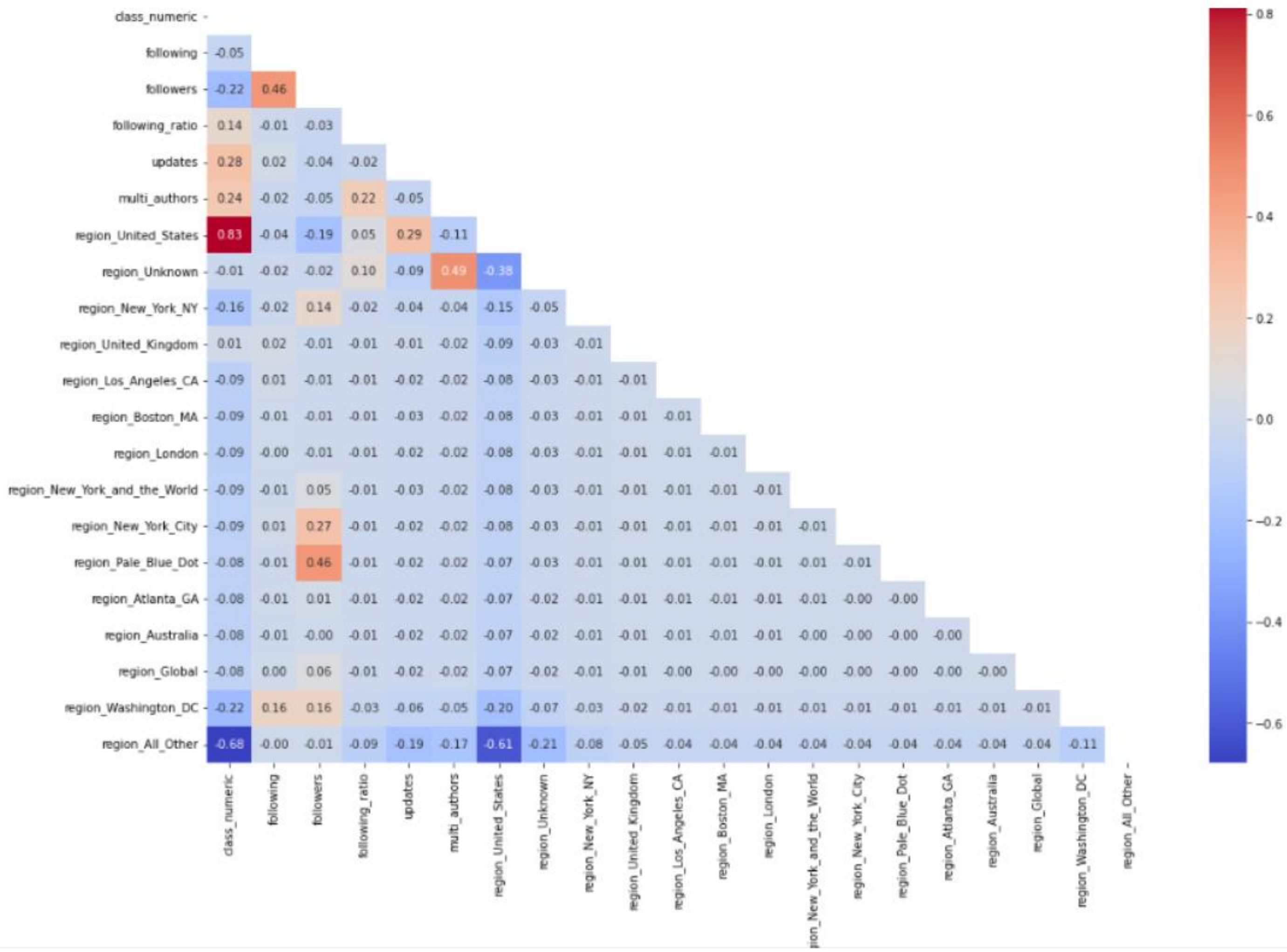


Figure 23 – Feature Correlation Map (Account Features) from Modeling Pipeline #2

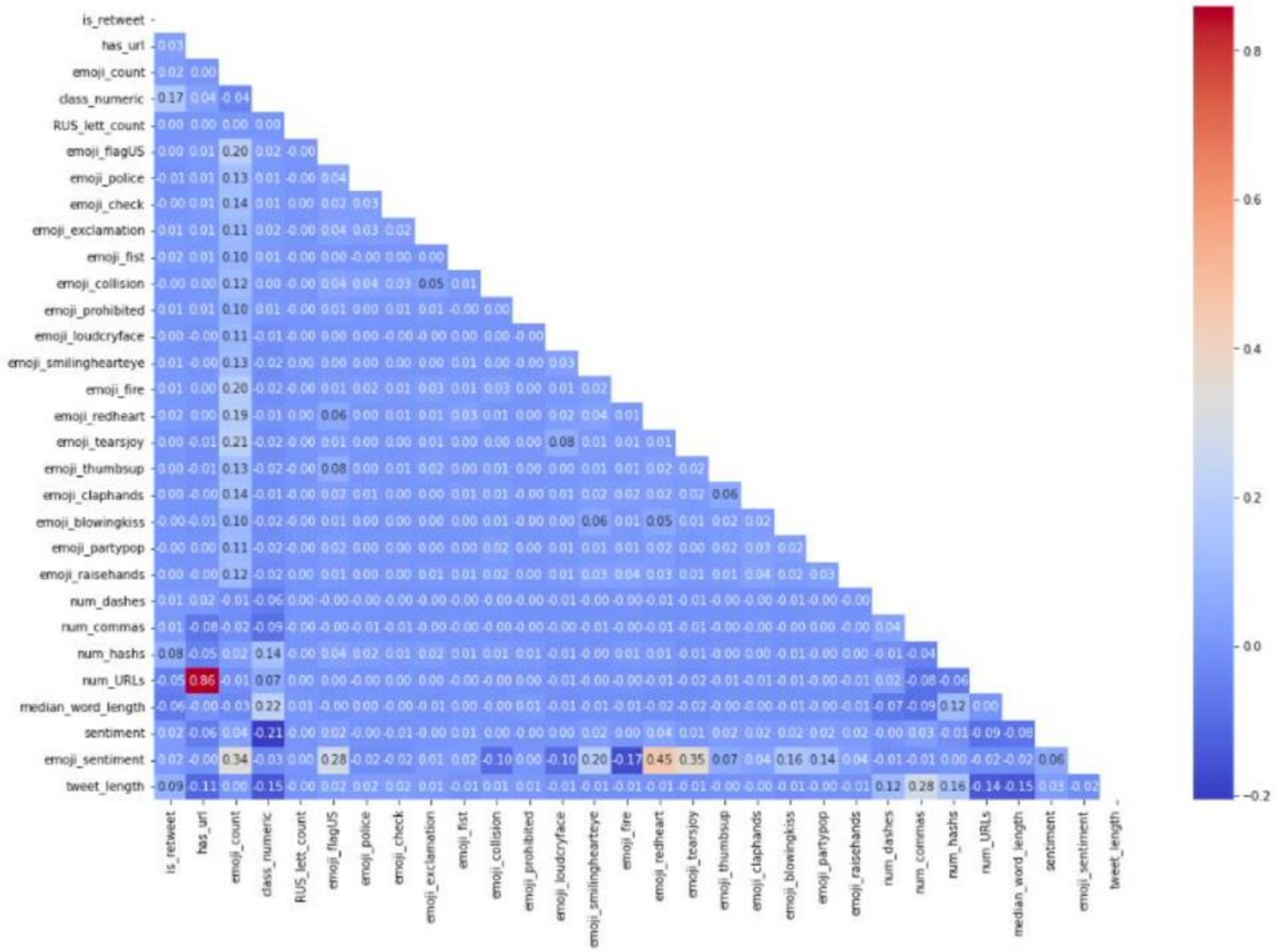


Figure 24 – Feature Correlation Heat Map (Tweet Text Features) from Modeling Pipeline #2

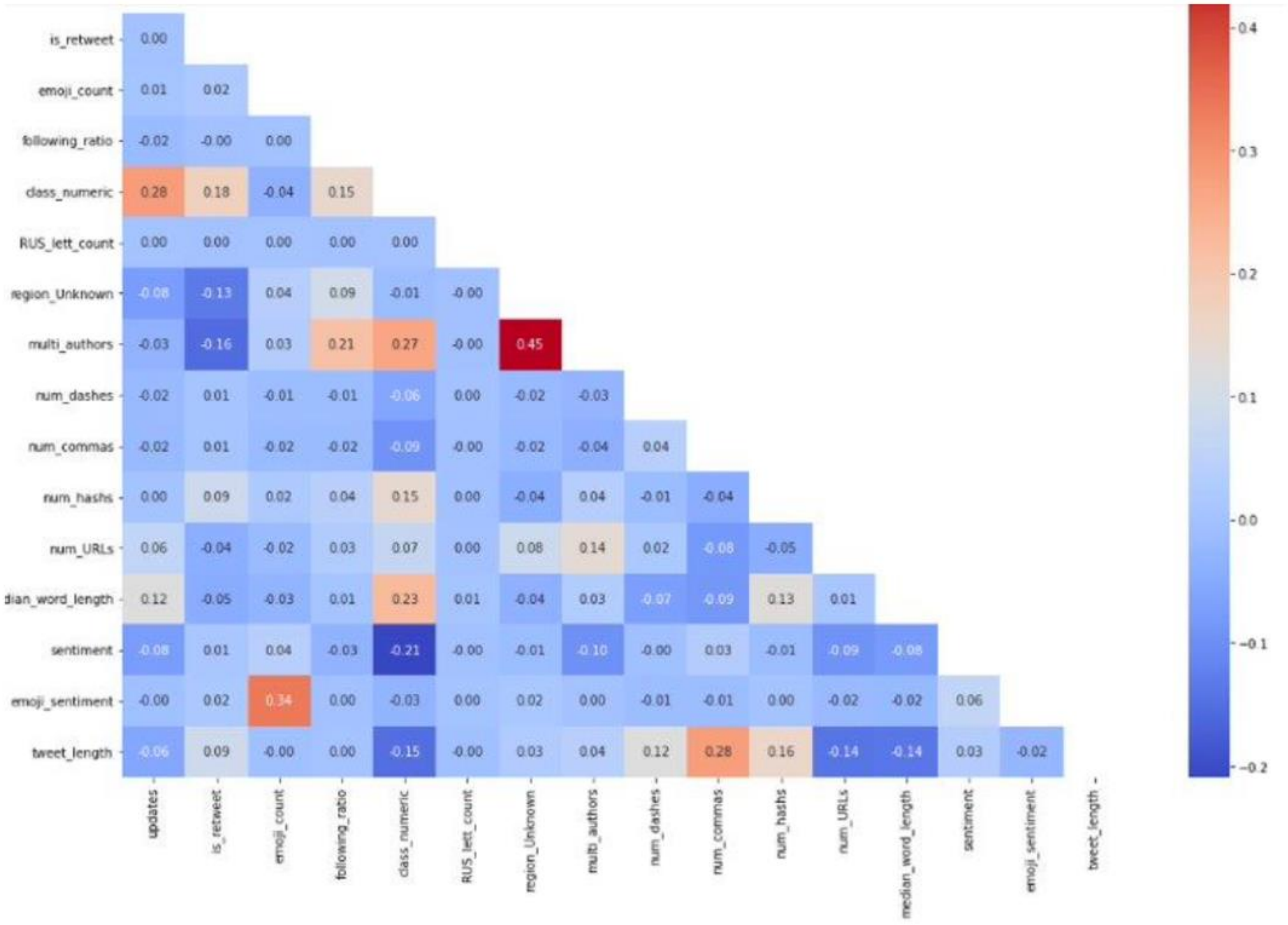


Figure 25 – Feature Correlation Heat Map (Reduced Dimensionality) from Modeling Pipeline #2

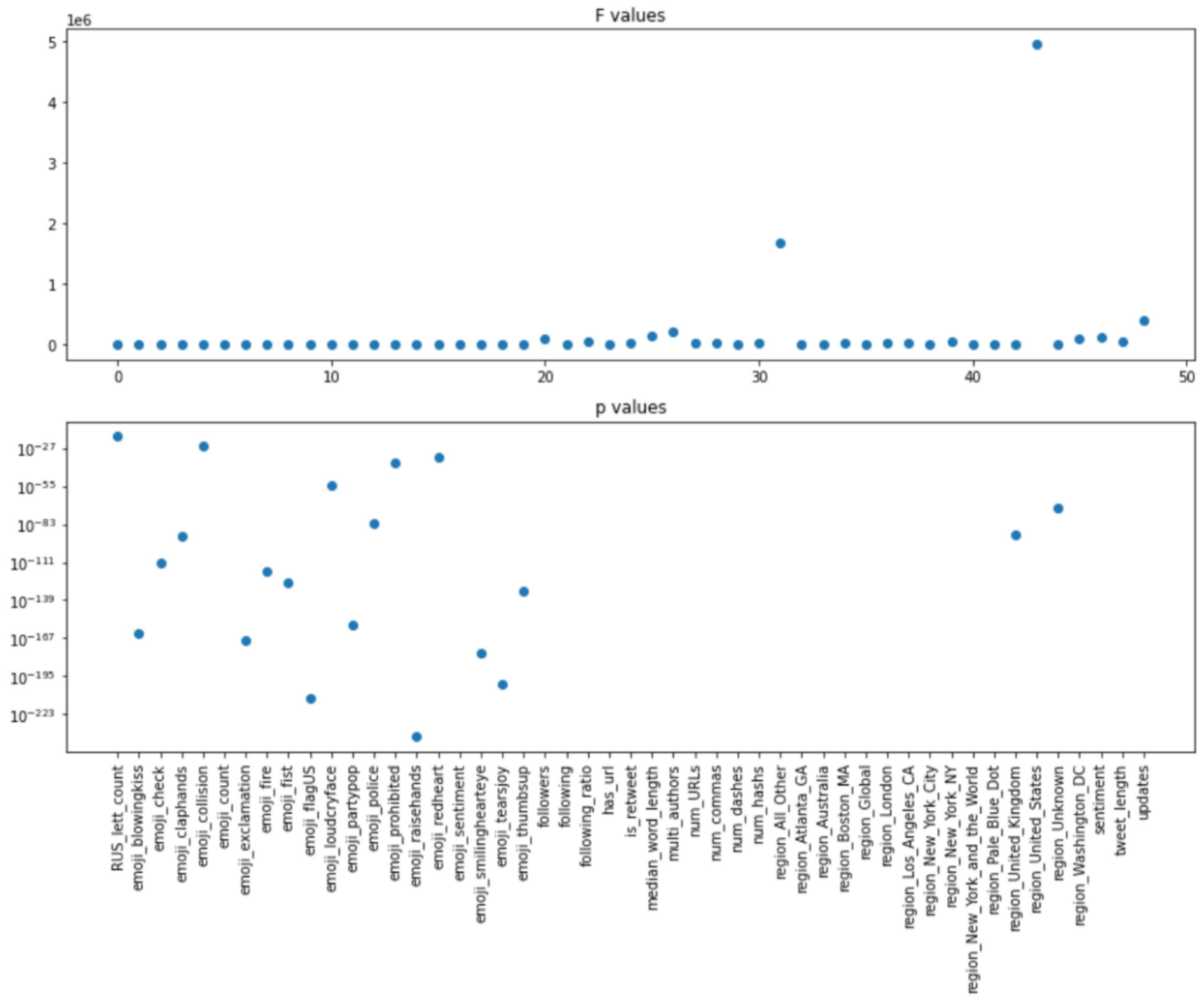


Figure 26 – F-values and P-values from Modeling Pipeline #2

Additional Tables

Table 5 – Data Dictionary

Feature	Description	Data Type	Notes
`external_author_id`	Author ID	Nominal	Unique identifier / primary key for twitter accounts
`author`	Author Twitter Handle	Categorical	The Twitter handle (i.e. username) of this tweet's author
`content`	Tweet Content	String Object	The text of the tweet itself, including hashtags, hyperlinks, mentions, etc.
`region`	Region	Categorical	A region classification. For troll tweets, determined by Social Studio
`language`	Language	Categorical	The language of the tweet
`following`	Number of Other Accounts Followed	Continuous Numerical	The number of other accounts that this tweet's account is following.
`followers`	Number of Other Accounts Following	Continuous Numerical	The number of other accounts that follow this tweet's account.
`updates`	Number of Update Actions	Continuous Numerical	A composite measure of public metrics for interaction by other users with a tweet.
`is_retweet`	Retweet	Binary	Binary indicator of whether or not the tweet is a retweet.
`account_category`	Account Category	Nominal	A categorization, primarily for differentiating types of troll accounts.
`tweet_id`	Tweet ID	Nominal	Unique identifier for each tweet, assigned by Twitter.
`tco1_step1`	First URL within Tweet Content	Nominal	The first URL redirected to after http://t.co/...
`data_source`	Data Source	Categorical	Either "Troll", "verified_user", or "verified_random"
`has_url`	Tweet Has URL	Binary	Does this tweet contain a URL?
`emoji_text`	Emoji Text	String / List of Strings	A list of natural language descriptions of each unique emoji used.
`emoji_count`	Number of Emoji	Continuous Numerical	The quantity of emoji used within the \`content\` of this tweet.
`publish_date`	Publish Date of Tweet	Interval	The original publish date of this tweet.
`RUS_lett_count`	Count of Russian alphabet letters	Continuous Numerical	The numeric quantity of Russian alphabet characters counted in a tweet.
`emoji_flagUS`	Emoji flag: United States US	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_police`	Emoji police car light 🚨	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_check`	Emoji check mark ✔	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_exclamation`	Emoji double exclamation mark ‼	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_fist`	Emoji fist ✊	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_collision`	Emoji collision 💥	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_prohibited`	Emoji prohibited 🚫	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_loudcryface`	Emoji loudly crying face 😭	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_smilinghearteye`	Emoji smiling face with heart-eyes 😍	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_fire`	Emoji fire 🔥	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_redheart`	Emoji red heart ❤️	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_tearsjoy`	Emoji face with tears of joy 😂	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_thumbsup`	Emoji thumbs up 👍	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_claphands`	Emoji clapping hands 👏	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_blowingkiss`	Emoji face blowing kiss 😘	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)

Table 5 – Data Dictionary

Feature	Description	Data Type	Notes
`emoji_partypop`	Emoji party popper 🎉	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`emoji_raisehands`	Emoji raising hands 🙌	Binary	Positive (1) if the tweet contains the described emoji, Negative (0)
`region_United_States`	United States	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_Unknown`	Unknown Region	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_New_York_NY`	New York NY	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_United_Kingdom`	United Kingdom	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_Los_Angeles_CA`	Los Angeles CA	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_Boston_MA`	Boston MA	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_London`	London	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_new_York_and_the_World`	New York and the World	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_New_York_City`	New York City	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_Pale_Blue_Dot`	Pale Blue Dot	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_Atlanta_GA`	Atlanta GA	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_Australia`	Australia	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_Global`	Global	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_Washington_DC`	Washington DC	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`region_All_Other`	All Other Regions	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`multi_authors`	Multiple Authors	Binary	One-Hot encoding of region column: (1) if region in "Description" value
`num_dashes`	Number of dashes	Continuous Numerical	Total number of dashes used in the tweet
`num_commas`	Number of commas	Continuous Numerical	Total number of commas used in the tweet
`num_hashs`	Number of hashtags	Continuous Numerical	Total number of hashtags used in the tweet
`num_URLs`	Number of URLs	Continuous Numerical	Total number of URLs used in the tweet
`median_word_length`	Median Word Length	Continuous Numerical	Median word length from the tweet
`cleaned_tweet`	Cleaned Tweet	String Object	Content column with cleaning function applied
`sentiment`	Sentiment Compound Score	Continuous Numerical	Compound sentiment score of the cleaned tweet
`emoji_sentiment`	Emoji Sentiment Compound Score	Continuous Numerical	Compound sentiment score of the emoji text
`tweet_length`	Length of Tweet	Continuous Numerical	Number of characters (length) of the tweet

Table 6 – List of accounts used for "verified_user" Twitter dataset (News Organizations and Prominent Users)

Account Type	Account Handle	Account Name	# Tweets Pulled	Timeframe
Verified News Organizations	`abcnews`	ABC News	239,165 *(Total across all news orgs)*	October 2013 October 2014 October 2015 October 2016 October 2017 (Applies to all news orgs listed)
	`AP`	Associated Press		
	`TheAtlantic`	The Atlantic		
	`business`	Bloomberg		
	`nytimes`	The New York Times (News)		
	`NPR`	NPR		
	`propublica`	ProPublica		
	`TIME`	TIME		
	`USATODAY`	USA Today		
	`axios`	Axios		
	`BBCWorld`	BBC		
	`csmonitor`	The Christian Science Monitor		
	`Forbes`	Forbes		
	`Newsweek`	Newsweek		
	`Reuters`	Reuters		
	`RealClearNews`	Real Clear Politics		
	`thehill`	The Hill		
	`WSJ`	The Wall Street Journal (News)		
	`FoxBusiness`	Fox Business		
	`TheIJR`	Independent Journal Review		
Account Type	**Account Handle**	**Account Name**	**# Tweets Pulled**	**Timeframe**
Verified Prominent Individuals	`MittRomney`	Mitt Romney	289	October 1, 2013 – Dec 31, 2017
	`MichelleObama`	Michelle Obama	70	
	`BarackObama`	Barack Obama	7,027	
	`HillaryClinton`	Hillary Clinton	9,825	
	`SarahPalinUSA`	Sarah Palin	8,566	
	`Mike_Pence`	Mike Pence	3,936	
	`elonmusk`	Elon Musk	3,115	
	`billgates`	Bill Gates	1,800	
	`CondoleezzaRice`	Condoleezza Rice	308	
	`DonaldJTrumpJr`	Donald J Trump Jr	10,362	

Table 7 – List of accounts used for "verified_user" Twitter dataset (Government Organizations)

Account Type	Account Handle	Account Name	# Tweets Pulled	Timeframe
Verified Government Organizations	`LibraryCongress`	Library of Congress	208,771	October 1, 2013 – Dec 31, 2017
	`CDCgov`	Centers for Disease Control		
	`HHSgov`	Dept of Health and Human Services		
	`StateDept`	Dept of State		
	`TheJusticeDept`	Dept of Justice		
	`US_FDA`	Food and Drug Administration		
	`FBI`	Federal Bureau of Investigation		
	`NIH`	National Institute of Health		
	`askTSA`	Transportation Safety Administration		
	`NSF`	National Science Foundation		
	`NASA`	National Aeronautics and Space Administration		
	`smithsonian`	Smithsonian Institute		
	`USDA`	Dept of Agriculture		
	`FCC`	Federal Communications Commission		
	`PeaceCorps`	Peace Corps		
	`EPAgov`	Environmental Protection Agency		
	`USGS`	US Geological Survey		
	`FEMA`	Federal Emergency Management Agency		
	`USARMY`	US Army		
	`USNWSgov`	National Weather Service		
	`DeptofDefense`	Dept. of Defense		

Table 8 – List of timeframes used for "verified_random" Twitter dataset

Account Type	Account Handle	Account Name	# Tweets Pulled	Timeframe
Randomized Tweets from Verified Accounts	*(various)*	*(various)*	95,432	2014-04-23 08:00 AM to 2014-04-23 12:00 PM
			99,835	2017-12-03 12:00 AM to 2017-12-03 04:00 AM
			130,685	2015-04-26 08:00 PM to 2015-04-27 12:00 AM
			114,964	2017-03-20 12:00 AM to 2017-03-20 04:00 AM
			93,315	2013-12-13 08:00 AM to 2013-12-13 12:00 PM
			180,535	2017-06-29 08:00 PM to 2017-06-30 12:00 AM
			183,010	2017-04-14 12:00 PM to 2017-04-14 04:00 PM
			76,445	2016-01-19 08:00 AM to 2016-01-19 12:00 PM
			96,440	2013-07-06 12:00 AM to 2013-07-06 04:00 AM
Total Tweets (Table 6, Table 7, and Table 8)			**1,563,587**	
Total Accounts (Table 6, Table 7, and Table 8)			**~140k**	

Table 9 – Feature subsets used in Modeling Pipelines #2 and #3

Account Derived Features

Feature Name	15 Features
following	
followers	
following_ratio	X
multi_authors	X
region_United_States	
region_Unknown	X
region_New_York_NY	
region_United_Kingdom	
region_Los_Angeles_CA	
region_Boston_MA	
region_London	
region_new_York_and_the_World	
region_New_York_City	
region_Pale_Blue_Dot	
region_Atlanta_GA	
region_Australia	
region_Global	
region_Washington_DC	
region_All_Other	

Text Derived Features

Feature Name	15 Features
updates	X
RUS_lett_count	X
emoji_flagUS	
emoji_police	
emoji_check	
emoji_exclamation	
emoji_fist	
emoji_collision	
emoji_prohibited	
emoji_loudcryface	
emoji_smilinghearteye	
emoji_fire	
emoji_redheart	
emoji_tearsjoy	
emoji_thumbsup	
emoji_claphands	
emoji_blowingkiss	
emoji_partypop	
emoji_raisehands	
num_dashes	X
num_commas	X
num_hashs	X
num_URLs	X
median_word_length	X
sentiment	X
emoji_sentiment	X
tweet_length	X
emoji_count	X
has_url	
is_retweet	X