# **George Mason University**

# **Social Media Disinformation Network**Analyzing Twitter Relationships in Python without an API | by Emre Rençberoğlu | Towards Data Science

**DAEN 690-002 Capstone (Fall 2020)**

**Team GAMUT**

**Rahul Sonti**

**Kausik Valeti**

**Bhagirath Reddy Vulupala**

**Nikhil Reddy Punreddy**

**Sindhuja Pulluri**

**December 6, 2020**

Contents

Abstract……………………………………………………………………………………………….1

[1 Introduction 2](#_Toc58092208)

[1.1 Background 2](#_Toc58092209)

[1.2 Problem Space 3](#_Toc58092210)

[1.3 Research 4](#_Toc58092211)

[1.4 Solution Space 7](#_Toc58092212)

[1.5 Project Objectives 7](#_Toc58092213)

[1.6 Primary User Story 7](#_Toc58092214)

[1.7 Product Vision – Sample scenarios 7](#_Toc58092215)

[ Scenario #1 7](#_Toc58092216)

[ Scenario #2 8](#_Toc58092217)

[2 Data Acquisition 8](#_Toc58092218)

[2.1 Overview 8](#_Toc58092219)

[2.2 Field Descriptions 8](#_Toc58092220)

[2.3 Data Context 8](#_Toc58092221)

[2.4 Data Conditioning 8](#_Toc58092222)

[2.5 Data Quality Assessment 9](#_Toc58092223)

[2.6 Other Data Sources 9](#_Toc58092224)

[3 Analytics and Algorithms 09](#_Toc58092225)

[4 Visualization 17](#_Toc58092226)

[5 Findings 24](#_Toc58092227)

[6 Summary 25](#_Toc58092228)

[7 Future work 25](#_Toc58092229)

[8 Appendix : Code References 26](#_Toc58092230)

[9 Appendix : Risk Section 27](#_Toc58092231)

[10 Appendix : Agile Development Workflow 28](#_Toc58092232)

[11 Appendix : References 29](#_Toc58092233)

Table of Figures:

[Figure 1 : Twitter Logo 2](#_Toc58092133)

[Figure 2: BERT Architecture 10](#_Toc58092134)

[Figure 3 : Mask Language Model 11](#_Toc58092135)

[Figure 4: Text Preprocessing 12](#_Toc58092136)

[Figure 5: Relationship of LSTM and Conveyor belts 13](#_Toc58092137)

[Figure 6: Architecture of LSTM 14](#_Toc58092138)

[Figure 7: Language distribution of turkey data 17](#_Toc58092139)

[Figure 8: Language distribution of train set 18](#_Toc58092140)

[Figure 9: Language distribution of test set 18](#_Toc58092141)

[Figure 10: Snippet of our data set 19](#_Toc58092142)

[Figure 11: Training and validation loss of our Model 19](#_Toc58092143)

[Figure 12: Model Performance 20](#_Toc58092144)

[Figure 13: Confusion Matrix of test set 20](#_Toc58092145)

[Figure 14: Language distribution of Russia data 21](#_Toc58092146)

[Figure 15: Language distribution of train data 21](#_Toc58092147)

[Figure 16: Language distribution of test data 22](#_Toc58092148)

[Figure 17: Snippet of our data 22](#_Toc58092149)

[Figure 18: Training and Validation loss of our Model 23](#_Toc58092150)

[Figure 19: Accuracy of our Model 23](#_Toc58092151)

[Figure 20: Confusion Matrix 24](#_Toc58092152)

Table of Tables:

[Table 1: Field Descriptions 8](#_Toc58092033)

[Table 2: Risk table 27](#_Toc58092034)

[Abstract](file:///C:\Users\Sindhuja\Downloads\DAEN%20690_GAMUT_Mid-Sprint3%20(1).docx#_Toc42001293)

Twitter is a social networking platform where many political thoughts and views are exchanged between users. Some of the users are, in fact, nation state actors – individuals having close links to the military, intelligence or state control apparatus of their country – who share fake news to engage in espionage, propaganda or disinformation campaigns. Twitter has already identified many of these accounts and banned them from Twitter for violating Twitter policies. Our main goal is to build a classification Natural Language Processing (NLP) model by learning disinformation and fake news patterns from tweets and to classify them either as “Disinformation” or “Others.” This study makes use of state-linked information operations (“IO”) data published by Twitter in June 2020 covering operations attributed to the People’s Republic of China (PRC), Russia, and Turkey. We narrowed our focus to the Turkish and Russian “IO” tweets. We also incorporated Twitter live stream data from the Twitter archives for the same period and isolated the banned Turkish and Russian accounts from the archived live stream data to create our “Others” category data. We used a Bidirectional Encoder Representation from Transformers (BERT) model and tested against archived Twitter tweets for the month period following the period of the training data. For the “Turkey” data our model predicted 29,443 out of 122,190 tweets as disinformation with an accuracy of 75.9%. For the same period Twitter banned only 26,259 disinformation tweets; so, based on our prediction model, it appears that Twitter may still be missing 3,184 additional disinformation tweets for that period. For the “Russia” data our model predicted 20,826 tweets out of 114,416 tweets as disinformation with an accuracy of 81.79%. For the same period Twitter banned only 3,163 disinformation tweets; so, based on our prediction model, it appears that Twitter may still be missing 17,663 additional disinformation tweets for that period.

# Introduction

## Background

Twitter is a 'microblogging' system that allows you to send and receive short posts called tweets. Tweets in general visible by publicly but senders can change the settings on who can view their tweets. They can be up to 140 characters long and can include links to relevant websites and resources. Twitter users follow other users. If you follow someone you can see their tweets in your twitter 'timeline'. You can choose to follow people and organizations with similar academic and personal interests to you. Using “retweet”, a tweet created by a user can be forwarded by another user to their feed.

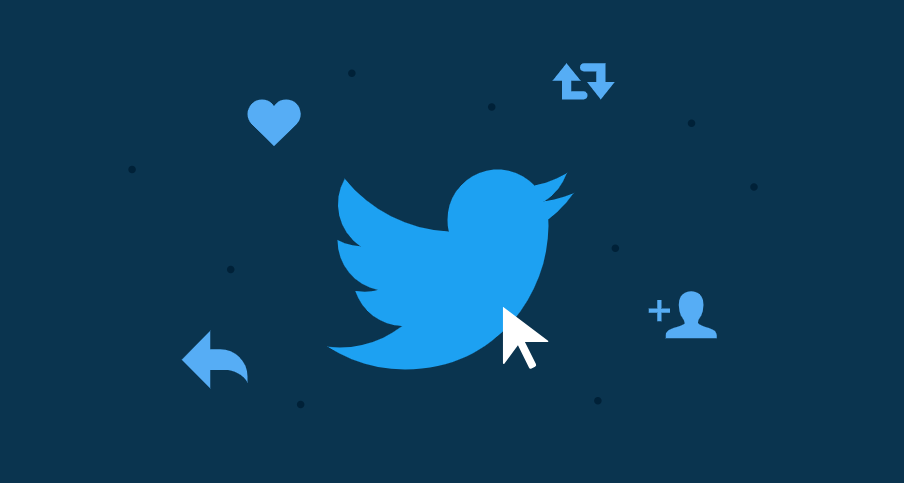


Figure : Twitter Logo

In January 2016, Twitter was sued by the widow of a U.S. man killed in the [2015 Amman shooting attack](https://en.wikipedia.org/wiki/2015_Amman_shooting_attack), claiming that allowing the [Islamic State of Iraq and the Levant](https://en.wikipedia.org/wiki/Islamic_State_of_Iraq_and_the_Levant) (ISIL) to continually use the platform, including direct messages in particular, constituted the [provision of material support to a terrorist organization](https://en.wikipedia.org/wiki/Providing_material_support_for_terrorism), which is illegal under U.S. federal law. Twitter disputed the claim, stating that "violent threats and the promotion of terrorism deserve no place on Twitter, and, like other social networks, our rules make that clear." The lawsuit was dismissed by the [United States District Court for the Northern District of California](https://en.wikipedia.org/wiki/United_States_District_Court_for_the_Northern_District_of_California), upholding the [Section 230](https://en.wikipedia.org/wiki/Section_230_of_the_Communications_Decency_Act) safe harbor, which dictates that the operators of an interactive computer service are not liable for the content published by its users. The lawsuit was revised in August 2016, providing comparisons to other telecommunications devices. On May 10, 2019, Twitter announced that they suspended 166,513 accounts for promoting terrorism in the July–December 2018 period, stating there was a steady decrease in terrorist groups trying to use the platform owing to its "zero-tolerance policy enforcement". According to [Vijaya Gadde](https://en.wikipedia.org/wiki/Vijaya_Gadde), Legal, Policy and Trust and Safety Lead at Twitter, there was a reduction of 19% terror related tweets from the previous reporting period (January–June 2018). Similarly, Twitter banned 7,000 accounts and limited 150,000 more that had ties to [QAnon](https://en.wikipedia.org/wiki/QAnon) on July 21, 2020. The bans and limits came after QAnon-related accounts began harassing other users through practices of swarming or brigading, coordinated attacks on these individuals through multiple accounts in the weeks prior. Those accounts limited by Twitter will not appear in searches nor be promoted in other Twitter functions. Twitter said they will continue to ban or limit accounts as necessary, with their support account stating "We will permanently suspend accounts Tweeting about these topics that we know are engaged in violations of our multi-account policy, coordinating abuse around individual victims, or are attempting to evade a previous suspension".

As Twitter is one of the social networking services, is used by millions of users around the world. On an average some billions of tweets posted per day. Considering few tweets of a specific period and on analyzing them, the tweets are categorized into different types based on the content like pointless tweets, conversational, self-promotion, spam news etc. Hashtags are used to discuss on a common issue with a “#” sign. A user can have multiple account or can create a fake account using the information of some other person. To overcome this issue, Twitter launched Verified Accounts program which verifies the identity of the user and a blue tick is shown on the profile which indicates the account is verified. Also, twitter bans some advertisements if they are violating the policies stated in the terms and conditions

A twitter transparency report is a statement issued on a regular basis by the organization, disclosing a variety of statistics related to requests for user data, records, or content. Transparency reports generally disclose how frequently and under what authority governments have requested or demanded data or records over a certain period. This form of corporate transparency allows the public to discern what private information governments have gained access to through search warrants and court subpoenas, among other methods.

Twitter data is the information collected by either the user, the access point, what is in the post and how users view or use your post. While this might sound somewhat vague, it is largely due to the massive amount of data that can be collected from a single Tweet. With this information, you can know demographics, total clicks on your profile or how many people saw your Tweet. In this scenario, we are focusing on the IO (Information Operations) Campaign data in which users from China, Turkey, and Russia have proposed false propaganda. We also used another class for the data other than the three countries already specified, namely “Others” which will be used for the classification purpose.

## Problem Space

To set the stage for the study, it is worth reviewing several areas of research that provide context for the study of disinformation in general and fake news in particular. One key piece of context is historical. Media and scholarly accounts have often emphasized the ways in which the fake news phenomenon is unprecedented. Yet the 2016 election is hardly the first time that false news stories, motivated by money or national ideology, have been aimed at the American public.

False news stories spread by The Associated Press helped lead to the inauguration of Rutherford B. Hayes as president and the end of post-Civil War Reconstruction. Most Americans are familiar with “yellow journalism,” sensational coverage that sold newspapers at the expense of factual accuracy at the turn of the 20th century. Yellow journalism strongly contributed to the start of the Spanish-American War and (arguably) the U.S. entry into World War I.15 Perhaps the most remarkable incident of fake news in American history—and far less known—was the massive, covert British propaganda effort to draw the U.S. into World War II. Run out of an office in Rockefeller Center in New York City, so-called British Security Coordination (BSC) involved as many as 3,000 British agents who manipulated U.S. news coverage on a massive scale. The effort paid friendly columnists and laundered (sometimes fake) British news stories through apparently unconnected outlets. The BSC campaign even developed the “game of Vik,” a large-scale campaign to anonymously harass Nazi sympathizers in the United States through tactics such as popping tires and putting rats in water tanks. Today we would call these types of acts trolling. During the Cold War, of course, the Soviet Union often targeted American audiences with false news stories. Declassified Russian documents show that by the early 1980s, the Soviets were spending more than $3 billion on external propaganda and influence campaigns, more than the U.S. was then spending on the National Security Agency. Soviet efforts routinely involved publishing fake news, including recruiting journalists as agents and publishing fabricated documents (often in troves of “leaked” genuine material). Fake news, then, is not unprecedented. Yet as the media environment and the political landscape have shifted, possibilities for fake news have mutated and metastasized. Aggregators and “content farms” have sprung up to produce low-quality, sensational, often misleading news stories framed to maximize clicks. New audience metrics and tools such as A/B testing may have encouraged a shift to sensational content. Changes in the media landscape coincided with broader polarization in the American public, with partisans showing increasing disdain for members of the other party. Social media is now the most important conduit of digital news, especially for many low-information voters.

A substantial and controversial literature has worried about online “echo chambers” or “filter bubbles,” in which individuals receive few political messages that contradict their prejudices, because of news self-selection, social homophily, or algorithms on big platforms such as Facebook. While several lines of empirical studies have complicated or directly challenged claims about strong filter bubbles, some research suggests that one-sided information flows produce bigger shifts in public opinion than balanced information flows. These lines of research have potentially important implications for our understanding of the impact of fake news on political attitudes and behavior. Partly in response to the digitization of the information landscape, there has been a wave of scholarship on how to correct misinformation. Much of this literature, unfortunately, has argued that false beliefs are often resistant to correction. Repeating false stories, even to debunk them immediately, might reinforce misperceptions (though scholarship on this point is conflicted). Even when members of the public do accept corrections, the initial false story can continue to affect attitudes. Research on misinformation has also emphasized the power of “social proof” in persuading the public to accept false information. People may be more apt to accept news stories as true when they come from friends and acquaintances and supposedly credible sources, and when these stories are more popular overall. Recent work has also found that repetition alone can make false news stories more believable. People are more accepting of the story the third or fourth time they are exposed to it, with familiarity increasing credibility. [1]

The specific problem we start with, we have an IO-Campaign tweet dataset with information approximately 32000 banned twitter accounts from China, Turkey and Russia. We added an extra class “Others” which is normal twitter tweet data which we extracted from the Twitter Archives by removing the banned users that are already present in the IO-Campaign data using SQL queries for User ID. We then develop a Bidirectional Encoder Representation from Transformers Model to train on this newly concatenated tweet data and use this model on unseen twitter tweet data to observe patterns in the tweets and classify them into the four classes namely: “China Disinfo Campaign”, “Russia Disinfo Campaign”, “Turkey Disinfo Campaign”, and “Others”.

## Research

The growing influence experienced by the propaganda of fake news is now cause for concern for all walks of life. Election results are argued on some occasions to have been manipulated through the circulation of unfounded and doctored stories on social media including microblogs such as Twitter. All over the world, the growing influence of fake news is felt on daily basis from politics to education and financial markets. The feat of accurately tracking the spread of fake messages and especially news content would be of interest to researchers, politicians, citizens as well as individuals all around the world. In addition, there is a growing and alarming use of social media for anti-social behaviors such as cyberbullying, propaganda, hate crimes, and for radicalization and recruitment of individuals into terrorism organizations such as ISIST ambuscio et al proposed a model for tracking the spread of hoaxes using four parameters: spreading rate, gullibility, probability to verify a hoax, and forgetting one's current belief.

A good example happened in the aftermath of Hurricane Sandy, where enormous amounts of fake and altered images were circulating on the internet. Gupta et al used a Decision Tree classifier to distinguish between fake and real images posted about the event Neural networks are a form of machine learning method that have been found to exhibit high accuracy and precision in clustering and classification of text . They also prove effective in the prompt detection of spatio-temporal trends in content propagation on social media. In this approach, we combine this with the efficiency of recurrent neural networks (RNN) in the detection and semantic interpretation of images. The approach of their work is twofold. First is the automatic identification of features within Twitter post without prior knowledge of the subject domain or topic of discussion using the application of a hybrid deep learning model of LSTM and CNN models. Second is the determination and classification of fake news posts on Twitter using both text and images

They implemented three deep neural network variants. The models applied to train the datasets include:

1. Long-Short Term Memory (LSTM)
2. LSTM with dropout regularization
3. LSTM with convolutional neural networks (CNN)

**Recurrent Neural Network (RNNs):**

This type of neural network has been shown to be effective in time and sequence-based predictions. Twitter posts can be likened to events that occur in time Recurrent Neural Networks were initially limited by the problem associated with the adjustment of weights over time. Several methods have been adopted in solving the vanishing gradient problem but can largely be categorized into two types, namely, the exploding gradient and the vanishing gradient. Solutions adopted for the former include truncated back propagation, penalties and gradient clipping (these resolve the exploding gradient problem), while the vanishing gradient problem has been resolved using dynamic weight initializations, the echo state networks (ESN) and Long-Short Term Memory (LSTMs). LSTMs will be the main focus of this work as they preserve the memory from the last phase and incorporate this in the prediction task of the neural network model.

**Incorporating Convolutional Neural Network:**

We posit that addition of the hybrid method would improve performance of the model and give much better results for the content based fake news detection. However, the hybrid implementation for this work so far involves a text-only approach.

**Selection of Training Parameters:**

The following hyper-parameters were optimized using a grid search approach and optimal values derived for the following batch size, epochs, learning rates, activation function and dropout regularization rate which is set at 20%

**Results:**

There deep learning model intuitively achieves 82% accuracy on the classification task in detecting fake news posts without prior domain knowledge of the topics being discussed. So far in the experiments completed, it is revealed that the plain vanilla LSTM model achieved the best performance in terms of precision, recall and F-Measure and an accuracy of 82%.LSTM method with dropout regularization performed the least in terms of the metrics adopted. This is likely a result of under fitting the model, in addition to the lack of enough training data, The LSTM-CNN hybrid model performed better than the dropout regularization model, with a 74% accuracy and an F-Measure of 39.7%. However insufficient training examples for the neural network model led to negative appreciation against the plain-vanilla LSTM model. Deep learning models such as CNN and RNN often require much larger datasets, and in some cases multiple layers of neural networks for the effective training of their models. In our case we have around **8 million** tweets.

Another model which is very useful Bidirectional Encoder Representations from Transformers BERT (Bidirectional Encoder Representations from Transformers), released in late 2018, is the model we will use in this tutorial to provide readers with a better understanding of and practical guidance for using transfer learning models in NLP. BERT is a method of pretraining language representations that was used to create models that NLP practitioners can then download and use for free. You can either use these models to extract high quality language features from your text data, or you can fine-tune these models on a specific task (classification, entity recognition, question answering, etc.) with your own data to produce state of the art predictions.

This post will explain how you can modify and fine-tune BERT to create a powerful NLP model that quickly gives you state of the art results.

**Advantages of Fine-Tuning:**

In this tutorial, we will use BERT to train a text classifier. Specifically, we will take the pre-trained BERT model, add an untrained layer of neurons on the end, and train the new model for our classification task. Why do this rather than train a train a specific deep learning model (a CNN, BiLSTM, etc.) that is well suited for the specific NLP task you need?

**Quicker Development:**

First, the pre-trained BERT model weights already encode a lot of information about our language. As a result, it takes much less time to train our fine-tuned model - it is as if we have already trained the bottom layers of our network extensively and only need to gently tune them while using their output as features for our classification task. In fact, the authors recommend only 2-4 epochs of training for fine-tuning BERT on a specific NLP task (compared to the hundreds of GPU hours needed to train the original BERT model or a LSTM from scratch!).

**Less Data:**

In addition, and perhaps just as important, because of the pre-trained weights this method allows us to fine-tune our task on a much smaller dataset than would be required in a model that is built from scratch. A major drawback of NLP models built from scratch is that we often need a prohibitively large dataset in order to train our network to reasonable accuracy, meaning a lot of time and energy had to be put into dataset creation. By fine-tuning BERT, we are now able to get away with training a model to good performance on a much smaller amount of training data.

**Better Results:**

Finally, this simple fine-tuning procedure (typically adding one fully-connected layer on top of BERT and training for a few epochs) was shown to achieve state of the art results with minimal task-specific adjustments for a wide variety of tasks: classification, language inference, semantic similarity, question answering, etc. Rather than implementing custom and sometimes obscure architectures shown to work well on a specific task, simply fine-tuning BERT is shown to be a better (or at least equal) alternative.

This shift to transfer learning parallels the same shift that took place in computer vision a few years ago. Creating a good deep learning network for computer vision tasks can take millions of parameters and be very expensive to train. Researchers discovered that deep networks learn hierarchical feature representations (simple features like edges at the lowest layers with gradually more complex features at higher layers). Rather than training a new network from scratch each time, the lower layers of a trained network with generalized image features could be copied and transferred for use in another network with a different task. It soon became common practice to download a pre-trained deep network and quickly retrain it for the new task or add additional layers on top - vastly preferable to the expensive process of training a network from scratch. For many, the introduction of deep pre-trained language models in 2018 (ELMO, BERT, ULMFIT, Open-GPT, etc.) signals the same shift to transfer learning in NLP that computer vision saw. [2]

## Solution Space

We are using BERT model tokenizer to generate attention masks, tokenids, labels we created a dataset called “pytorch”, we defined SoftMax function along with BERT pretrained model to classify tweets into China, Russia, Turkey and Others. In our future work for image segmentation we would be using CNN, Fast R-CNN, YOLO etc. models for the image classification part of the project.

## Project Objectives

We aim at developing a model which classifies the users based on their tweets into four different classes namely: Russia, China, Turkey, and Others. We will have an idea about the dispersion of fake news across the Twitter platform and will be able to achieve a goal of filtering these fake news and accounts that are spreading this false propaganda. There has been a massive increase in the fake new in the past decade and this problem can be mitigated using our models in this project. The targeted groups range from Government Organizations to Non- Profit Organizations who plan to reduce the violations that are caused due to the fake news.

## Primary User Story

“Based on the user context and value proposition, we developed the following primary user story to guide our project:

“As a User, I want to submit the twitter data to train it with Recurring Neural Network model and apply on test twitter data to classify the tweets into four classes which are “China”, “Russia”, “Turkey” and “Others””

## Product Vision – Sample scenarios

* For: Classifying Data based on Tweets and Images by users
* Who: Twitter Users
* Is a: Classification Model
* That: Classifies users among 4 different classes
* Our product: Computer Vision & Image classification

### Scenario #1

One typical example is a post found on numerous right-wing Facebook groups. The post used old and mislabeled photos of injured police officers in order to claim that caravan migrants were behaving violently toward law enforcement. The false post was shared thousands of times.

### Scenario #2

Other situation where this project can be of prime importance is when users take to Twitter to propose fake news about Corona Virus and its vaccine information across the countries Russia, China, and Turkey. This can be identified from our model and can be used to filter out this data and help the necessary sources to mitigate this fake news.

# Data Acquisition

## Overview

We have datasets of size around 90GB which consists of tweets and images of different sizes with china consisting of 31GB media size and 73.2 MB of tweet information, Turkey 821 GB media size and 5 GB of tweet information and Russia with 108GB media size and 353 MB of tweet information. After data cleaning and data preprocessing we have dataset of size 267MB which we are using this to train our Model.

## Field Descriptions

| Field | Type | Description |
| --- | --- | --- |
| Tweet Id | Varchar | The unique id for tweet provided by twitter to fetch it easily from the data. This id can be used to identify the tweet, the time, and date. |
| Tweet text | Varchar | The content that user posts on twitter which explains what the user is trying to imply. |
| Location | String | The location of the user at that point of time when the user posted the tweet. |

Table : Field Descriptions

## Data Context

The Data is a combination of four different datasets combined using SQL which include tweets of banned users by three countries Russia, Turkey and china and another dataset has original tweets. The dataset of Russia, Turkey and china is obtained from the twitter transparency report website, In the website there are a combination of JSON files which has all the twitter tweets data that includes the information of banned users of the above-mentioned countries. These JSON files are converted into Excel using SQL and from this data, on removing the data of the banned user details of the above-mentioned countries we get another excel data which is named as “Others”. These four datasets namely Russian, Turkey, China and Others combined to a single dataset which we are using as our final data for this Project.

## Data Conditioning

To optimize the data we are working on, we developed instances which use Zzip package for the conversion of. json format data files into .csv format. The Twitter Archives has data in subfolders of files in. json format. These files needed to be converted and combined to get an optimized dataset to apply our desired classification model. An addition to Zzip, Glob package was used to retrieve the files that have a pattern from the Archives. Glob works for. json format files and is used to observe and retrieve pattern identified files. Once the data was retrieved from the Archives, as well as the Transparency report, we appended by data by rows using SQL functions by adding another attribute “Region”: “Others” to the Archive data. So, with the help of the above-mentioned libraries, and platform, we optimized the data for implementing the Classification model.

## Data Quality Assessment

* **Completeness:** The dataset needs preprocessing that must be done to eliminate the null records or the missing data
* **Uniqueness:** All the attributes in the dataset are unique and of differentiable datatypes.
* **Accuracy:** The data is true without noise because the datasets maintains the quality to perform analysis and is likely to be the most accurate data based on the case studies by Twitter
* **Atomicity:** The data is updatable at any point of time to add more records and delete the records present in the data. We have a flexibility to modify the data as per the needs and utilize it for feeding to the Classification model.
* **Conformity:** The datasets acquired are qualified to perform analysis on because these datasets are used to generate global reports on various related issues.
* **Overall Quality:** Twitter publishes this data every year for stating the privacy of their data and the security of their customers. So, the quality of data is definitely worth performing analysis on.

## Other Data Sources

* Our aim is to classify tweets for Russia, Turkey, China and others. Twitter archive has provided data which is enough for our problem since the aim is to classify tweets.

# Analytics and Algorithms

For the data we have we are going to use BERT (Bidirectional Encoder Representation from Transformers) because unlike RNN (LSTM and GRU), it does not take the sequential approach and rather goes with the Attention-based approach. It learns contextual relations between words (or sub-words) in a text and classifies the data into the respective classes. The Transformer encoder reads the entire sequence of words at once. Therefore, it is considered bidirectional, or rather non-directional. This characteristic allows the model to learn the context of a word based on all its surroundings (left and right of the word). Since our data output is to divide into four classes.

**BERT Model:**

BERT stands for Bidirectional Encoder Representations from Transformers. It is intended to pre-train deep bidirectional portrayals from unlabeled content by together conditioning on both left and right context. Subsequently, the pre-trained BERT model can be calibrated with only one extra output layer to make best in class models for a wide scope of NLP tasks. It is based on a transformer architecture.

BERT is a method of pretraining language representations that was used to create models that NLP practitioners can then download and use for free. You can either use these models to extract high quality language features from your text data, or you can fine-tune these models on a specific task (classification, entity recognition, question answering, etc.) with your own data to produce state of the art predictions. Advantages of Fine Tuning are Quicker Development, Less Data, Better results. [2]

BERT is also pre-trained on a large corpus of unlabeled text including the entire Wikipedia (that is 2,500 million words!) and Book Corpus (800 million words). This pre-training step is the magic behind BERT’s success. This is because as we train a model on a large text corpus, our model starts to pick up the deeper and intimate understandings of how the language works. This knowledge is much useful for any NLP task.

BERT is a deeply bidirectional model which means that BERT learns information from both the left and right side of a token’s context during the training phase. The bidirectionality of a model is important for truly understanding the meaning of a language. BERT considers both the left and right context before making a prediction. The most impressive aspect of BERT is we can fine-tune it by adding just a couple of additional output layers to create state-of-the-art models for a variety of NLP tasks. BERT has significantly altered the NLP landscape. It has inspired many recent NLP architectures, training approaches and language models, such as Google’s TransformerXL, OpenAI’s GPT-2, XLNet, ERNIE2.0, RoBERTa, etc. [3]

**BERT’s Architecture:**

The BERT architecture builds on top of Transformer. We currently have two variants available:

* BERT Base: 12 layers (transformer blocks), 12 attention heads, and 110 million parameters
* BERT Large: 24 layers (transformer blocks), 16 attention heads and, 340 million parameters [4]

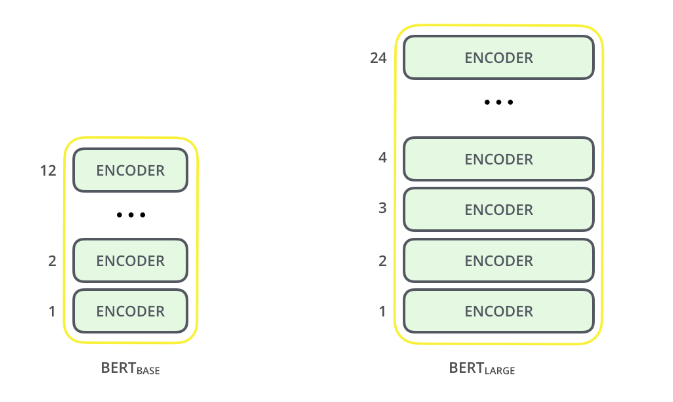


Figure : BERT Architecture

**BERT’S Transformer:**

The Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence aligned RNNs or convolution. The transformers play a major role in capturing relationships and sequence of words in sentences which is important for a machine to understand a natural language. Here the transduction means the conversion of input sequences into output sequences. The transformers handle the dependencies between input and output with attention and recurrence completely. All the transformer layers are Encoder-only blocks.

BERT uses a multi-layer bidirectional Transformer encoder. Its self-attention layer performs self-attention in both directions. It uses bidirectionality by pre-training on a couple of tasks such as Masked Language Model and Next Sentence Prediction. [5]

**BERT Pre-Training Tasks:**

BERT is pre-trained using the two unsupervised prediction tasks which are Masked Language Modeling (MLM) and Next Sentence Prediction.

* The Masked Language Model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context. Unlike left-to-right language model pre-training, the MLM objective allows the representation to fuse the left and the right context, which allows us to pre-train a deep bidirectional Transformer



Figure : Mask Language Model

* Next Sentence Prediction understands the relationship between sentences when performed on a task. In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence. [6]

**Text Preprocessing:**

The developers behind BERT have added a specific set of rules to represent the input text for the model. Many of these are creative design choices that make the model even better.

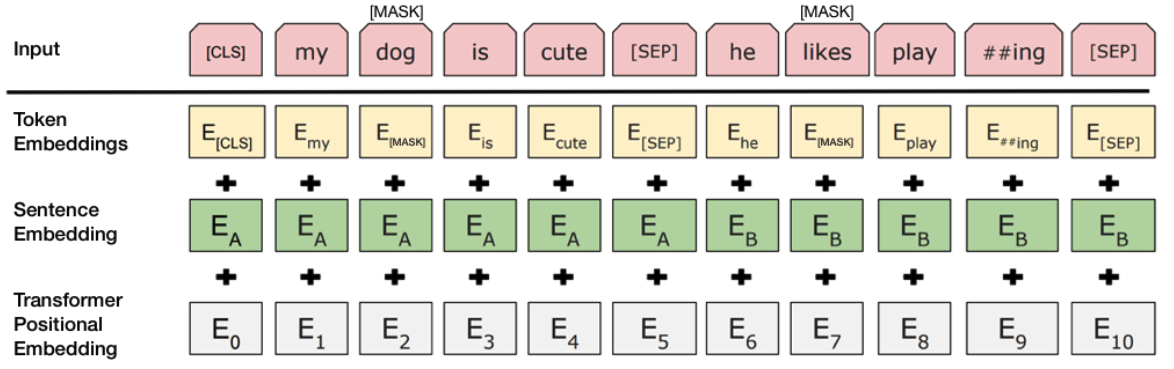


Figure : Text Preprocessing

Every input embedding is a combination of 3 embeddings:

* **Position Embeddings:** BERT learns and uses positional embeddings to express the position of words in a sentence. These are added to overcome the limitation of Transformer which, unlike an RNN, is not able to capture “sequence” or “order” information.
* **Segment Embeddings:** BERT can also take sentence pairs as inputs for tasks (Question-Answering). That’s why it learns a unique embedding for the first and the second sentences to help the model distinguish between them. In the above example, all the tokens marked as EA belong to sentence A (and similarly for EB).
* **Token Embeddings:** These are the embeddings learned for the specific token from the Word Piece token vocabulary. For a given token, its input representation is constructed by summing the corresponding token, segment and position embeddings. [4]

**Long Short-Term Memory (LSTM):**

Turns out that an RNN doesn’t do so. In order to add a new information, it transforms the existing information completely by applying a function. Because of this, the entire information is modified, overall, i. e. there is no consideration for ‘important’ information and ‘not so important’ information.

LSTMs on the other hand, make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

These dependencies can be generalized to any problem as:

1. The previous cell state (i.e. the information that was present in the memory after the previous time step)
2. The previous hidden state (i.e. this is the same as the output of the previous cell)
3. The input at the current time step (i.e. the new information that is being fed in at that moment)

Another important feature of LSTM is its analogy with conveyor belts! That’s right! Industries use them to move products around for different processes. LSTMs use this mechanism to move information around. We may have some addition, modification or removal of information as it flows through the different layers, just like a product may be molded, painted or packed while it is on a conveyor belt. The following diagram explains the close relationship of LSTMs and conveyor belts.

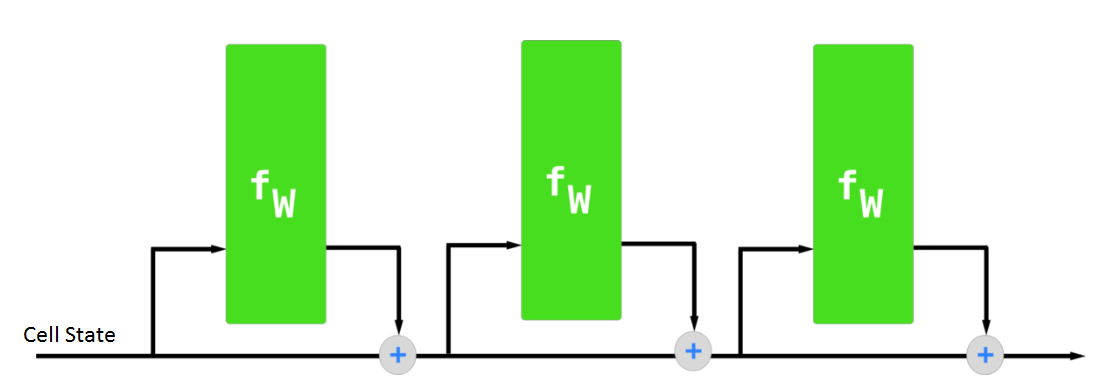


Figure : Relationship of LSTM and Conveyor belts

**Architecture of LSTM:**

The functioning of LSTM can be visualized by understanding the functioning of a news channel’s team covering a murder story. Now, a news story is built around facts, evidence and statements of many people. Whenever a new event occurs you take either of the three steps.

Let’s say, we were assuming that the murder was done by ‘poisoning’ the victim, but the autopsy report that just came in said that the cause of death was ‘an impact on the head’. Being a part of this news team what do you do? You immediately forget the previous cause of death and all stories that were woven around this fact.

What, if an entirely new suspect is introduced into the picture. A person who had grudges with the victim and could be the murderer? You input this information into your news feed, right?

Now all these broken pieces of information cannot be served on mainstream media. So, after a certain time interval, you need to summarize this information and output the relevant things to your audience. Maybe in the form of “XYZ turns out to be the prime suspect.”.

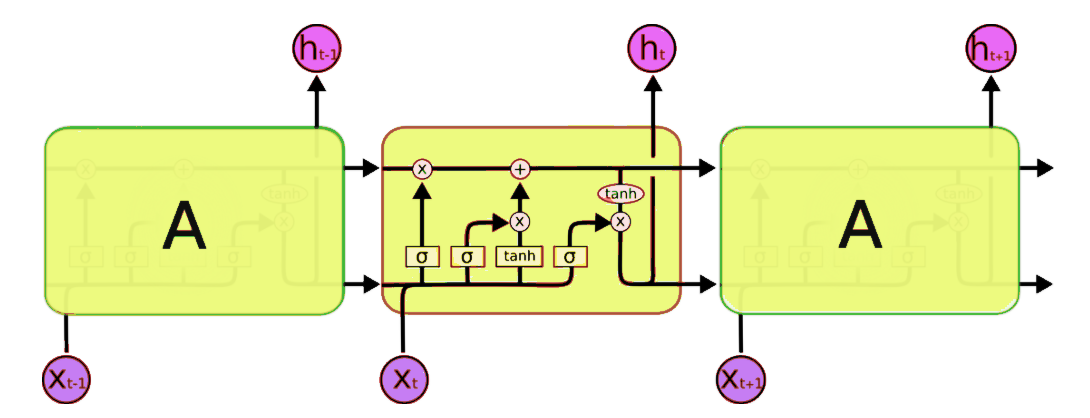


Figure : Architecture of LSTM

Now, this is nowhere close to the simplified version which we saw before but let me walk you through it. There are two states that are being transferred to the next cell; the cell state and the hidden state. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called gates. [7]

**Text generation using LSTM:**

1. Importing dependencies
2. Preparing dataset
3. Defining the LSTM Model
4. Fitting the Model and Generating characters.

**Support Vector Machine:**

A support vector machine (SVM) is a supervised [machine learning](http://www.monkeylearn.com/machine-learning/) model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they’re able to categorize new text.

So, you’re working on a text classification problem. You’re refining your training data, and maybe you’ve even tried stuff out using Naive Bayes. But now you’re feeling confident in your dataset and want to take it one step further. Enter Support Vector Machines (SVM): a fast and dependable classification algorithm that performs very well with a limited amount of [data to analyze](http://www.monkeylearn.com/data-analysis/). Perhaps you have dug a bit deeper, and ran into terms like *linearly separable*, *kernel trick* and *kernel functions*. But fear not! The idea behind the SVM algorithm is simple and applying it to natural language classification doesn’t require most of the complicated stuff.

**The Kernel trick:**

we found a way to classify nonlinear data by cleverly mapping our space to a higher dimension. However, it turns out that calculating this transformation can get computationally expensive: there can be a lot of new dimensions, each one of them possibly involving a complicated calculation. Doing this for every vector in the dataset can be a lot of work, so it’d be great if we could find a cheaper solution.

And we’re in luck! Here’s a trick: SVM doesn’t need the actual vectors to work its magic, it actually can get by only with the [dot products](https://en.wikipedia.org/wiki/Dot_product) between them. This means that we can sidestep the expensive calculations of the new dimensions!

That’s it! That’s the kernel trick, which allows us to sidestep a lot of expensive calculations. Normally, the kernel is linear, and we get a linear classifier. However, by using a nonlinear kernel (like above) we can get a nonlinear classifier without transforming the data at all: we only change the dot product to that of the space that we want and SVM will happily chug along. Note that the kernel trick isn’t actually part of SVM. It can be used with other linear classifiers such as logistic regression. A support vector machine only takes care of finding the decision boundary.

**SVM with Natural Language Classification:**

So, we can classify vectors in multidimensional space. Great! Now, we want to apply this algorithm for text classification, and the first thing we need is a way to transform a piece of text into a vector of numbers so we can run SVM with them. In other words, which features do we have to use in order to classify texts using SVM? The most common answer is word frequencies, [just like we did in Naive Bayes](https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/#feature-engineering). This means that we treat a text as a bag of words, and for every word that appears in that bag we have a feature. The value of that feature will be how frequent that word is in the text.

This method boils down to just counting how many times every word appears in a text and dividing it by the total number of words. So, in the sentence “All monkeys are primates but not all primates are monkeys” the word *monkeys* have a frequency of 2/10 = 0.2, and the word *but* has a frequency of 1/10 = 0.1. For a more advanced alternative for calculating frequencies, we can also use [TF-IDF](https://en.wikipedia.org/wiki/Tf%E2%80%93idf). Now that we’ve done that, every text in our dataset is represented as a vector with thousands (or tens of thousands) of dimensions, everyone representing the frequency of one of the words of the text. Perfect! This is what we feed to SVM for training. We can improve this by using preprocessing techniques, like [stemming](https://en.wikipedia.org/wiki/Lemmatisation), removing [stop words](https://en.wikipedia.org/wiki/Stop_words), and using [n-grams](http://sebastianraschka.com/Articles/2014_naive_bayes_1.html#n-grams).

**Choosing a Kernel function:**

Now that we have the feature vectors, the only thing left to do is choosing a kernel function for our model. Every problem is different, and the kernel function depends on what the data looks like. In our example, our data was arranged in concentric circles, so we chose a kernel that matched those data points.

Taking that into account, what’s best for natural language processing? Do we need a nonlinear classifier? Or is the data linearly separable? It turns out that it’s best to stick to a linear kernel. Why?

Back in our example, we had two features. Some real uses of SVM in other fields may use tens or even hundreds of features. Meanwhile, NLP classifiers use thousands of features, since they can have up to one for every word that appears in the training data. This changes the problem a little bit: while using nonlinear kernels may be a good idea in other cases, having this many features will end up making nonlinear kernels overfit the data. Therefore, it’s best to just stick to a good old linear kernel, which actually results in the best performance in these cases. Now the only thing left to do is training! We have to take our set of labeled texts, convert them to vectors using word frequencies, and feed them to the algorithm — which will use our chosen kernel function — so it produces a model. Then, when we have a new unlabeled text that we want to classify, we convert it into a vector and give it to the model, which will output the tag of the text. [8]

**Deploying BERT Model in AWS:**

1. **Training Model:**

we use the [PyTorch-Transformers library](https://pytorch.org/hub/huggingface_pytorch-transformers/), which contains PyTorch implementations and pretrained model weights for many NLP models, including BERT. Their training script should save model artifacts learned during training to a file path called model\_dir, as stipulated by the Amazon Sage Maker PyTorch image. Upon completion of training, Amazon Sage Maker uploads model artifacts saved in model\_dir to Amazon S3 so they are available for deployment. Because PyTorch-Transformer isn’t included natively in Amazon Sage Maker PyTorch images, we have to provide a requirements.txt file so that Amazon Sage Maker installs this library for training and inference. A requirements.txt file is a text file that contains a list of items that are installed by using pip install. You can also specify the version of an item to install. [9]

We use Amazon Sage Maker to train and deploy a model using our custom PyTorch code [10]. The Amazon Sage Maker Python SDK makes it easier to run a PyTorch script in Amazon Sage Maker using its PyTorch estimator. After that, we can use the Sage Maker Python SDK to deploy the trained model and run predictions. For more information about using this SDK with PyTorch, see [Using PyTorch with the Sage Maker Python SDK](https://sagemaker.readthedocs.io/en/stable/using_pytorch.html). To start, we use the PyTorch estimator class to train our model. When creating the estimator, we make sure to specify the following:

* entry\_point – The name of the PyTorch script
* source\_dir – The location of the training script and requirements.txt file
* framework\_version: The PyTorch version we want to use

The PyTorch estimator supports multi-machine, distributed PyTorch training. To use this, we just set train\_instance\_count to be greater than 1. Our training script supports distributed training for only GPU instances. After creating the estimator, we call fit (), which launches a training job. We use the Amazon S3 URIs we uploaded the training data to earlier. [11]

1. **Deploying Model:**

After training our model, we host it on an Amazon Sage Maker endpoint by calling deploy on the PyTorch estimator. The endpoint runs an Amazon Sage Maker PyTorch model server. We need to configure two components of the server: model loading and model serving. We implement these two components in our inference script train\_deploy.py. The complete file is available in the [GitHub repo](https://github.com/aws-samples/amazon-sagemaker-bert-pytorch). model\_fn () is the function defined to load the saved model and return a model object that can be used for model serving. The Sage Maker PyTorch model server loads our model by invoking model\_fn: input\_fn () deserializes and prepares the prediction input. In this use case, our request body is first serialized to JSON and then sent to model serving endpoint. Therefore, in input\_fn (), we first deserialize the JSON-formatted request body and return the input as a torch. tensor, as required for BERT, predict\_fn () performs the prediction and returns the result. We take advantage of the prebuilt Amazon Sage Maker PyTorch image’s default support for serializing the prediction result.

# Visualization

Turkey:

The below graph is the Language distribution of Turkey data of 1st week of June 2019, the tweets are in 97 languages, we can see from the graph that Turkish language tweets (80,963 tweets) are predominant one in the distribution.

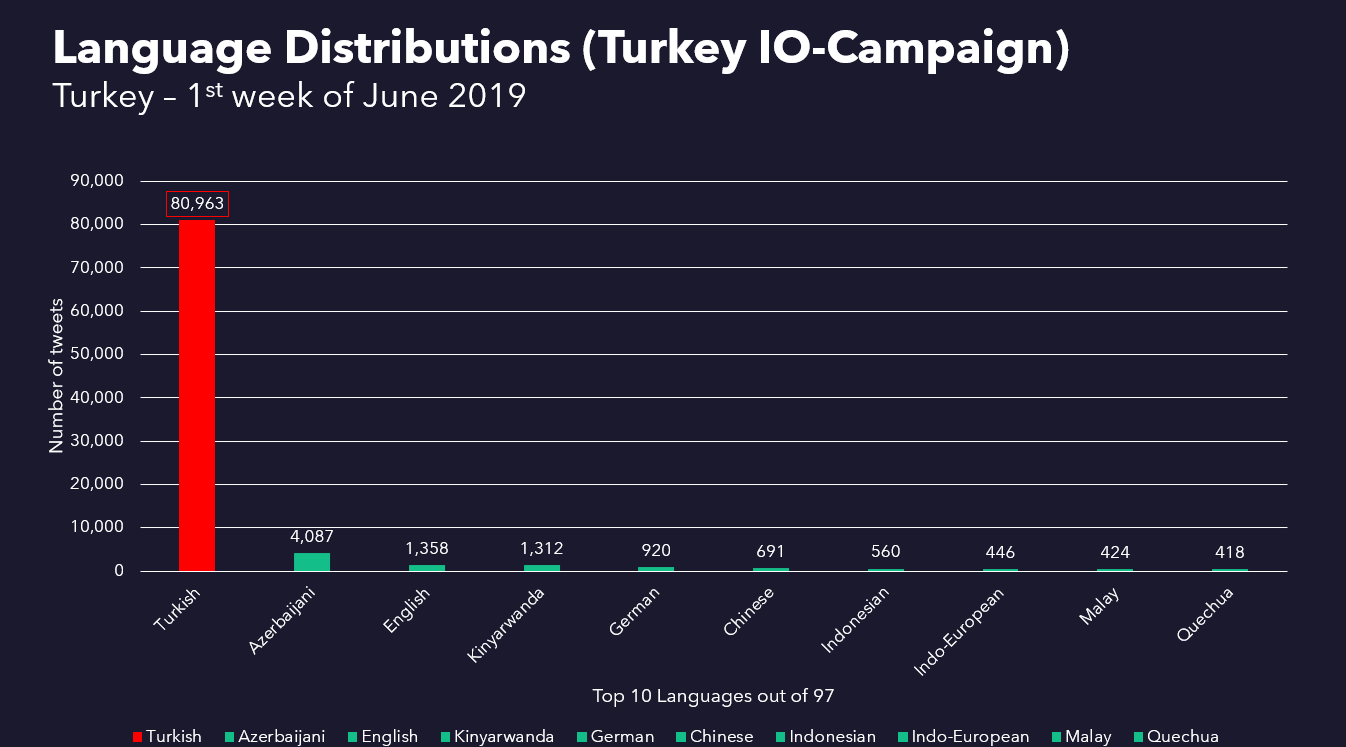


Figure : Language distribution of turkey data

The below picture shows the Language distribution of our training set (June 2019) i.e. Turkey-Others data. We can see that out of 97 languages in the data English language tweets are predominant in number (2,149,671) followed by Japanese tweets

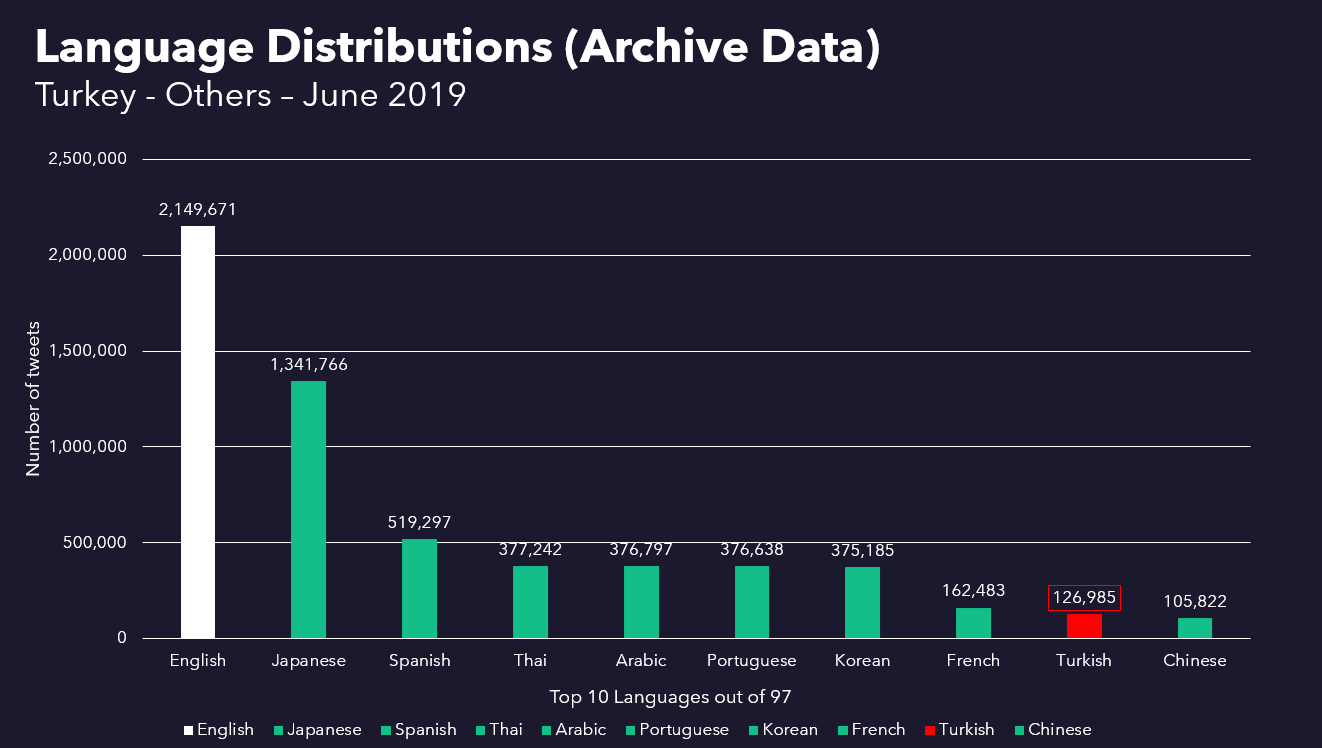


Figure : Language distribution of train set

The below picture shows the Language distribution of our test set (1st week of July 2019) i.e. Turkey-Others data. We can see that out of 97 languages in the data English language tweets are predominant in number (7,534,567) followed by Japanese tweets

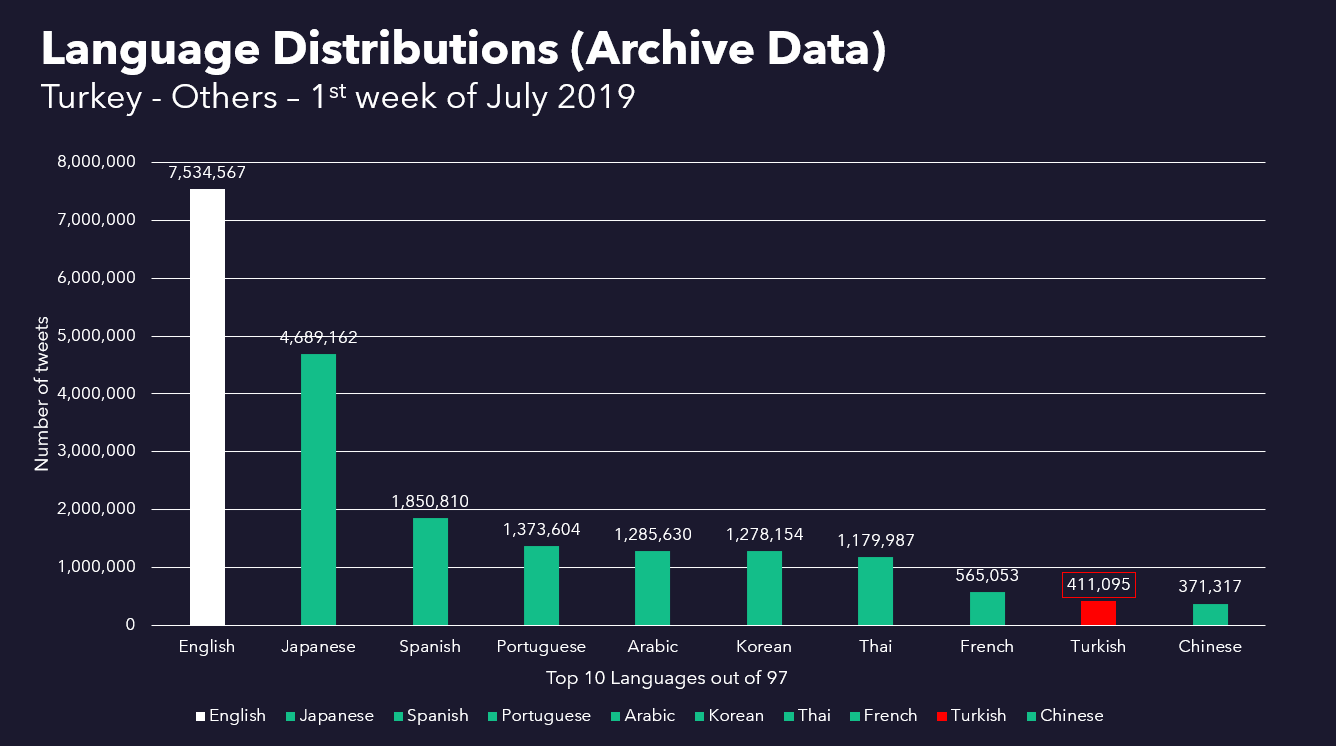


Figure : Language distribution of test set

The below picture shows the snippet of our dataset

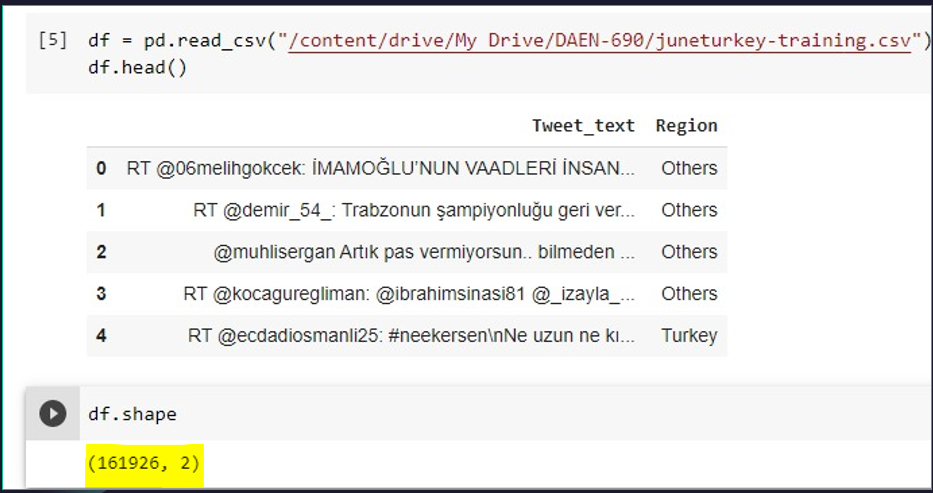


Figure : Snippet of our data set

The below Picture shows the Training and validation loss of our model for 3 epochs.

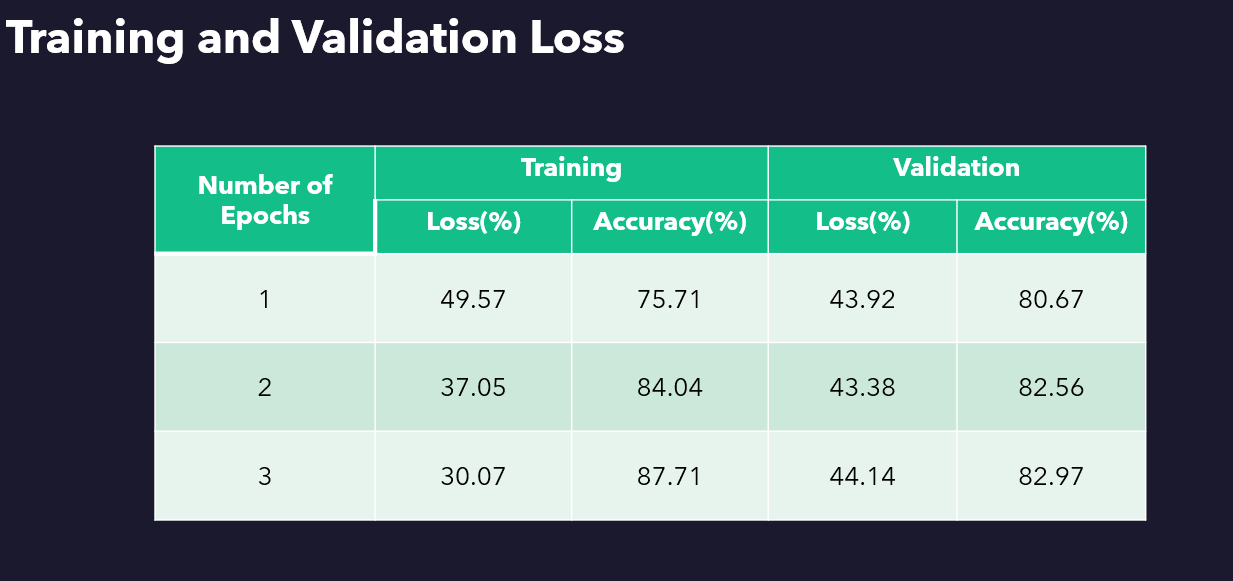


Figure : Training and validation loss of our Model

The below pictures show the accuracy of our model, we got an accuracy of 89.40% for our test data

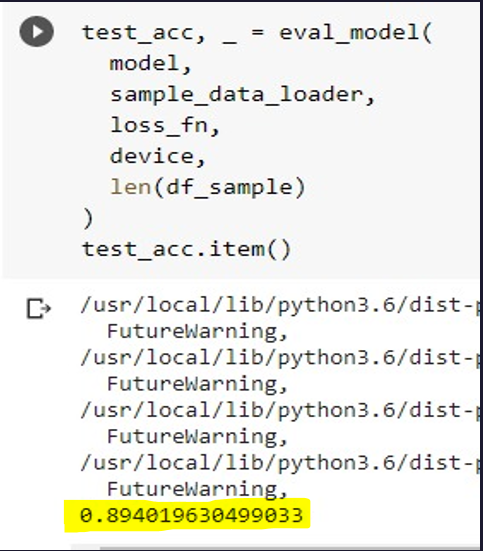


Figure : Model Performance

The below picture shows the confusion matrix, we can observe that number of tweets that others are classified correctly (367527 tweets). It predicted most of the labels correctly as our model performed better.

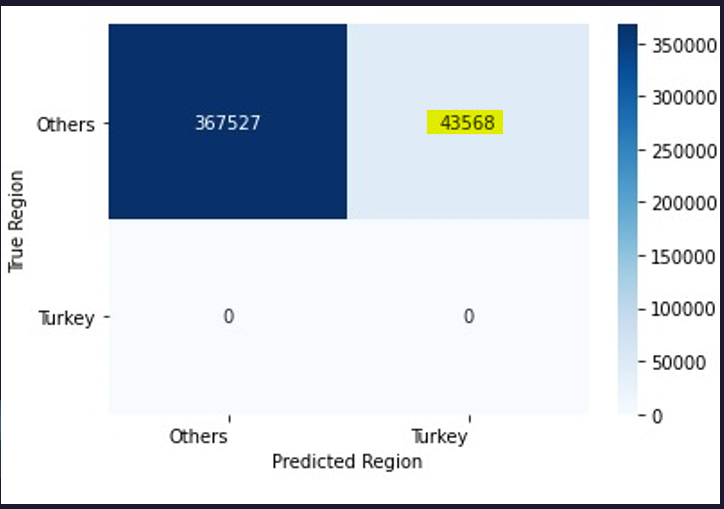


Figure : Confusion Matrix of test set

**Russia:**

The below graph is the Language distribution of Russia data of 1st week of June 2019, the tweets are in 97 languages, we can see from the graph that Russian language tweets (12,159 tweets) are predominant one in the distribution.

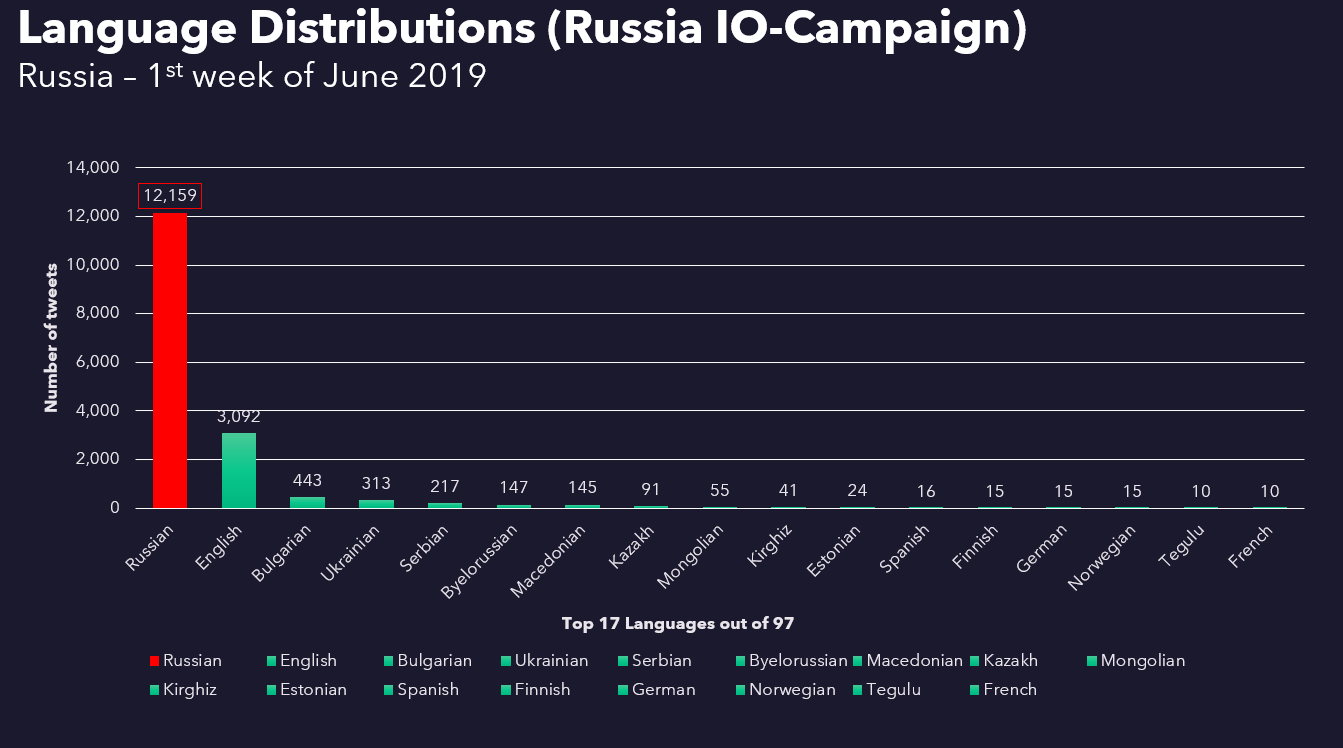


Figure : Language distribution of Russia data

The below picture shows the Language distribution of our training set (June 2019) i.e. Russia-Others data. We can see that out of 97 languages in the data English language tweets are predominant in number (2,151,393) followed by Japanese tweets

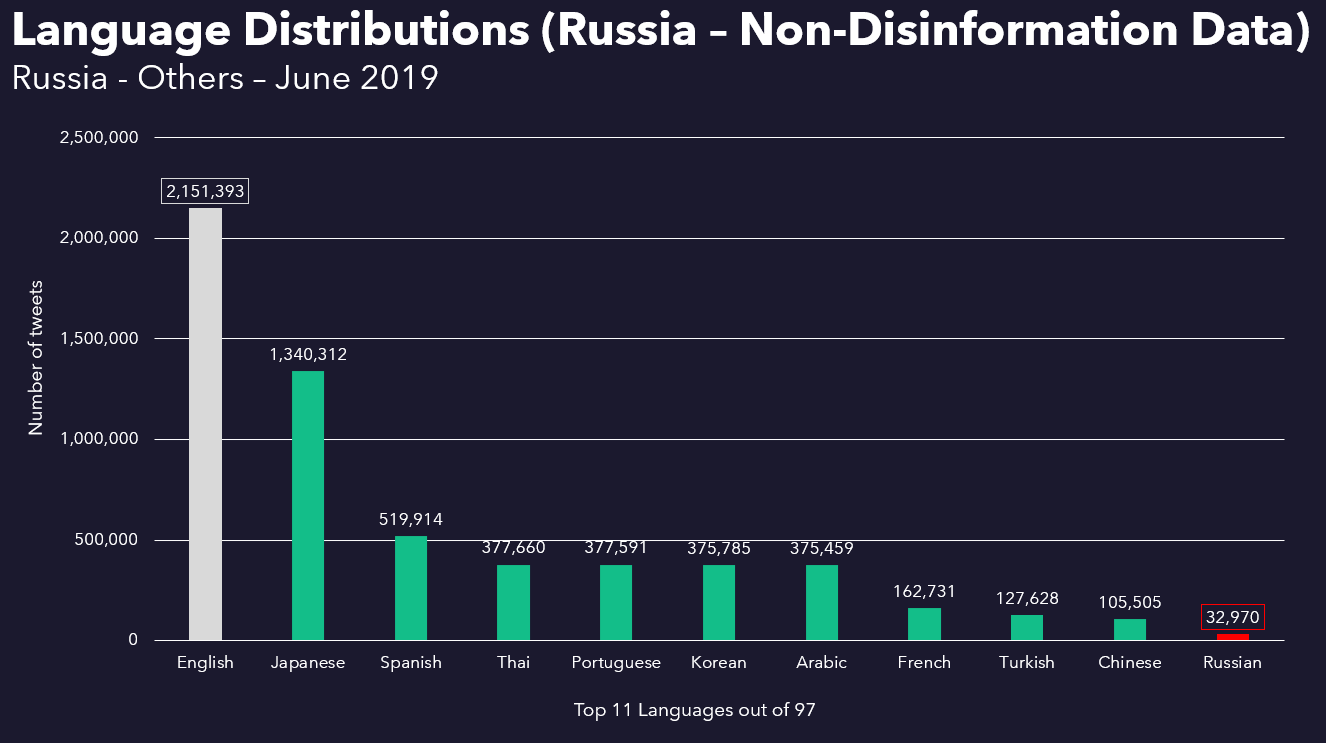


Figure : Language distribution of train data

The below picture shows the Language distribution of our test set (1st week of July 2019) i.e. Turkey-Others data. We can see that out of 97 languages in the data English language tweets are predominant in number (7,534,574) followed by Japanese tweets

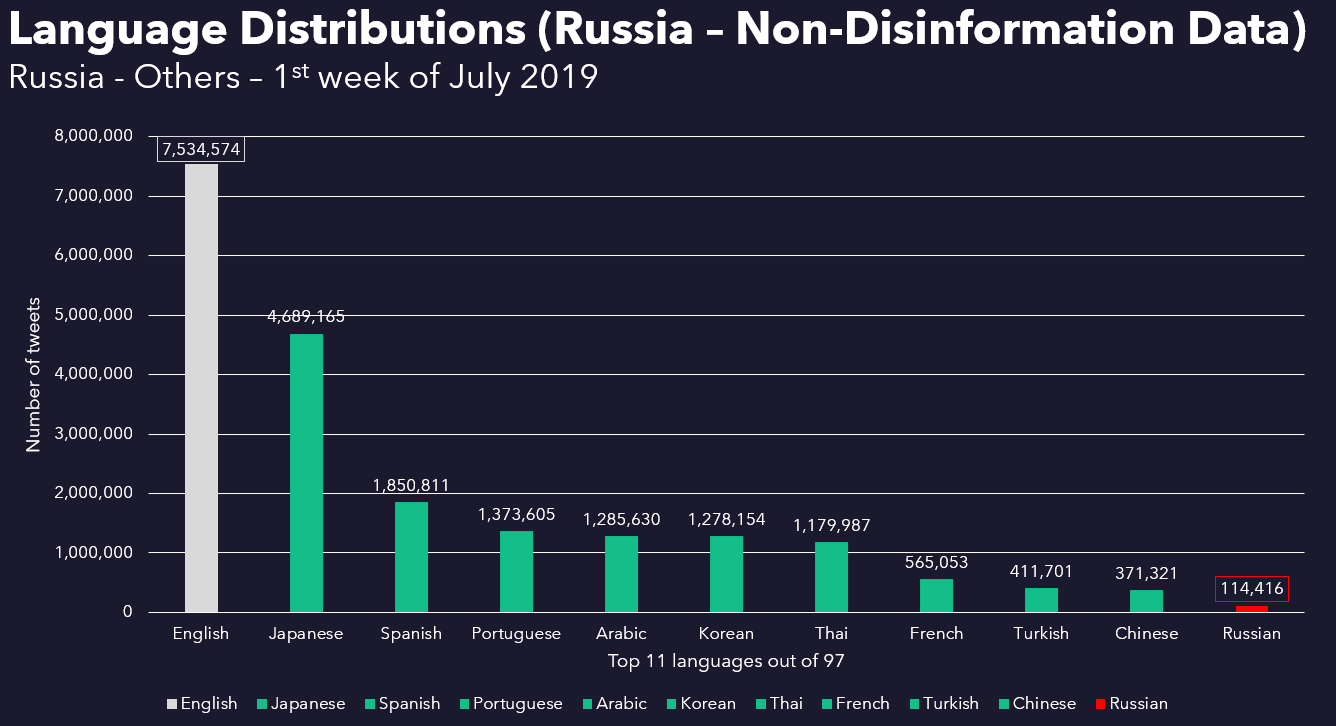


Figure : Language distribution of test data

The below picture shows the snippet of our dataset

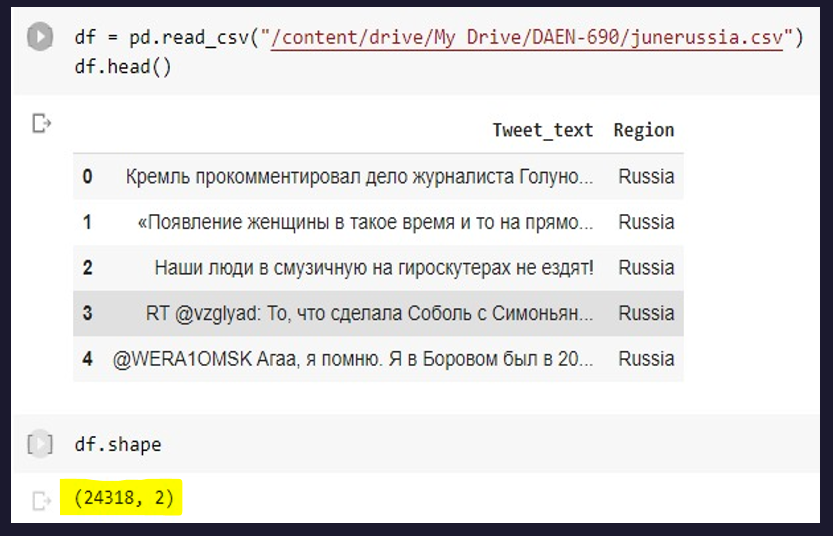


Figure : Snippet of our data

The below Picture shows the Training and validation loss of our model for 3 epochs.

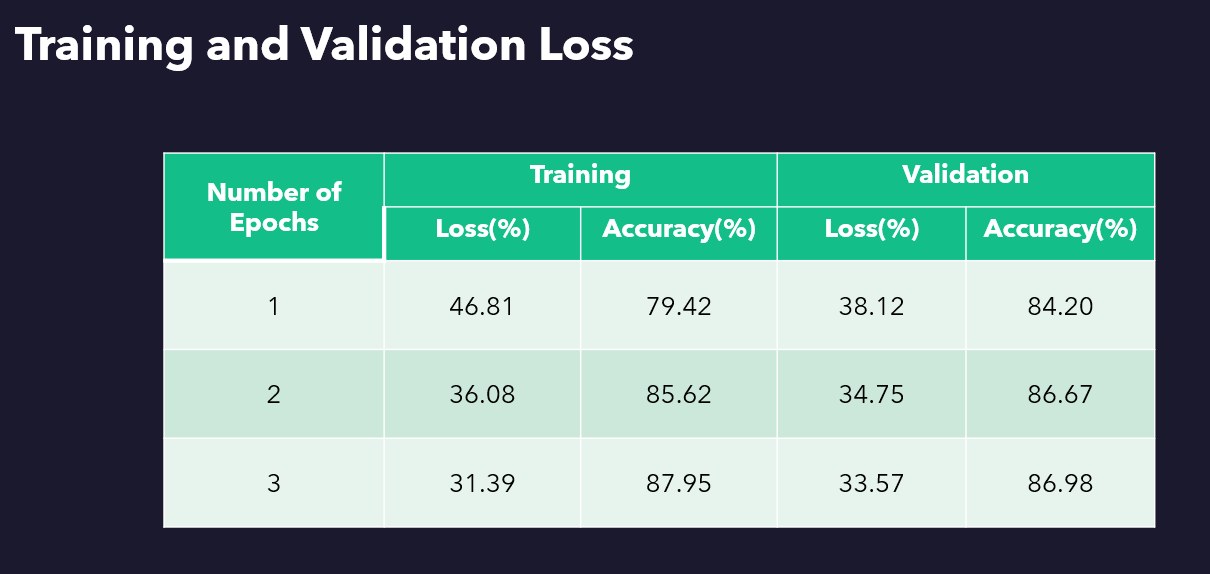


Figure : Training and Validation loss of our Model

The below pictures show the accuracy of our model, we got an accuracy of 81.79% for our test data

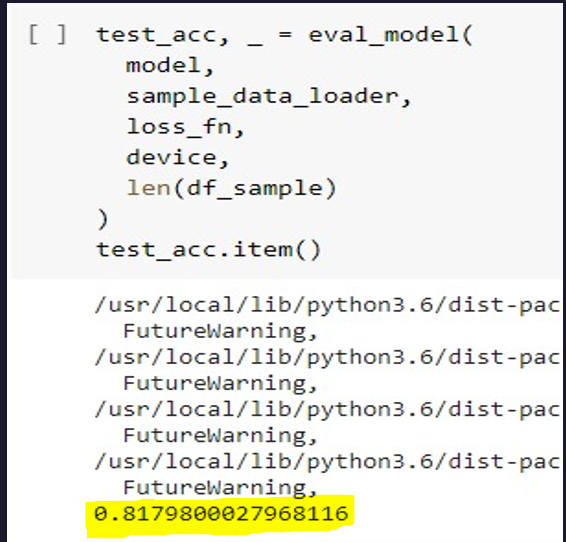


Figure : Accuracy of our Model

The below picture shows the confusion matrix, we can observe that number of tweets that others are classified correctly (93590 tweets). It predicted most of the labels correctly as our model performed better.

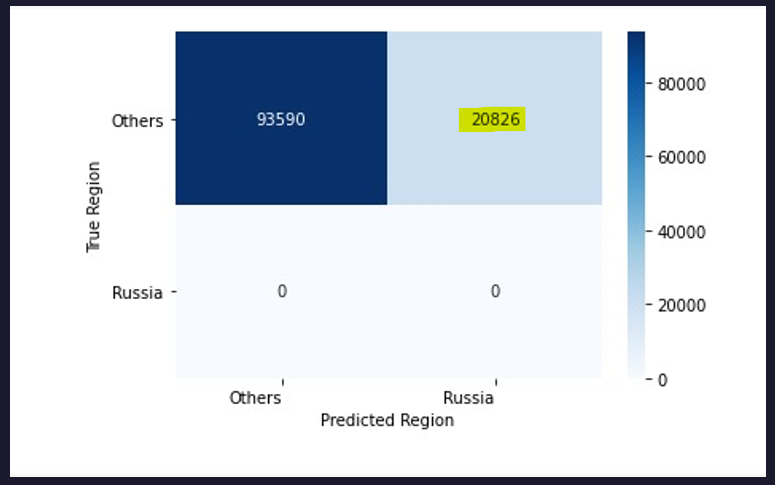


Figure : Confusion Matrix

# Findings

Twitter’s principle of transparency is to improve public understanding of inauthentic influence campaigns, in order it made its archives of Tweets publicly available. Twitter’s responsibility is to protect the integrity of the public conversation including through the timely disclosure of information about attempts to manipulate Twitter to influence elections and other civic conversations by foreign or domestic state-backed entities. Twitter has banned around 11,000 accounts from Russia and Turkey countries which are running fake campaigns and damaging the integrity of twitter. We developed a model to check whether twitter missed any fake accounts in that particular time period from livestream data and we built a model that may be predicting better than what twitter has been using as we got a more number of fake accounts and tweets than twitter. We have achieved this with BERT Model by learning pattern from banned tweet accounts. We used Spyder IDE(Integrated development environment) for the data pre-processing and used SQLite for getting the desired data by removing the screen names that are already available in the transparency report from the Archive data and merging them. After getting our desired data, we converted the tables in SQL into csv files and used this data for doing language distribution plots in Jupyter notebook . After that we indexed our data for just Turkish tweets and Russian tweets and created csv files for training and testing purposes and stored them on our google drive for colab purpose. We developed our BERT pretrained classification model on google colab for each of Turkish Disinformation – Non-Disinformation and Russian Disinformation – Non-Disinformation and stored the results back in the google drive.

# Summary

Using a Bidirectional Encoder Representation from Transformers (BERT) model, with the “Turkey” & “Russia” information operations and “Others” live stream archive category data for training, we tested this model against archived Twitter tweets for the month time period following the time period of the training data. Our model predicted 43,568 tweets as “Turkey” disinformation out of 411,095 tweets with an accuracy of 89.4%. For the same time period Twitter banned only 26,259 disinformation tweets. Based on our prediction model it appears that Twitter may still be missing 17,309 information operations tweets for that time period, similarly Our model predicted 20,826 tweets as “Russia” disinformation out of 114,416 tweets with an accuracy of 81.79%. For the same time period Twitter banned only 3,163 disinformation tweets. Based on our prediction model it appears that Twitter may still be missing 17,663 information operations tweets for that time period, and we are satisfied with results and this model can be used for all languages and many disinfo networks. This model can be used for identifying any kind of disinformation pattern to classify the text labels.

# Future work

We did not have sufficient data available for china to discover findings as we had for Turkey and Russia. In the event that Twitter gives more banned accounts data, it would have been better for analysis on China Disinformation. Moreover, when compared to Turkey’s data, we did not have sufficient data for Russia, to compare and find better in-depth insights for all the three countries and about their disinformation campaigns. We developed the BERT (Bidirectional Encoder Representation from Transformers) model since it is a Transformer-based machine learning technique for natural language processing pre-training developed by Google.

The questions however pertaining the concrete pattern identification without performing Natural Language Processing explicitly persist. How well is BERT pre-trained model (or) any other state-of-the-art models working for identifying the patterns in the data without the language identification itself? Also, how well can a model like BERT (or) any other deep-learning model understand the context in multiple languages and answers the forthcoming questions about prediction or classification? In our case, in the event that the model might have identified the patterns in the data (we assume the model did) for each of Turkey, Russia, and China, is it possible to provide functionality to identify the accounts that proposed the false propaganda to make the job of reporting these accounts much easier? Also, disinformation can be proposed in many formats apart from text, like videos which possibly say a lot more about the campaign than plain text. How well can these state-of-the-art models perform on identifying the disinformation from data like this, and has Twitter investigated the possibility of banning the accounts which posted videos proposing disinformation?

An interesting approach to finding disinformation from images can be developed using advanced Deep-Learning models like YOLO(You-Only-Look-Once), RCNN(Region-Bound-Convolutional-Neural-Networks), etc. But the problem lies in identifying the language the disinformation is proposed. All these advanced models work very well with a context in English. There are approaches to translate the text from another language to English using pre-developed APIs, but there is a possibility of losing a lot of information when translating. How well can a model be developed for multi-language context detection in images, and process this context for identifying the patterns for further use in campaigns like Country-Disinformation? We look forward to many more advancements in the field of multi-language classifiers for text, and images to avoid NLP for easier convention and seeking the desired discoveries.

# Appendix : Code References

**BERT Model code reference:**

<https://github.com/TeamGAMUT/Social_Media_Disinformation_Network>

GitHub, Inc. is a subsidiary of Microsoft which provides hosting for software development and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration, and wikis for every project. Headquartered in California, it has been a subsidiary of Microsoft since 2018.

We hosted our project code which includes SQL queries for indexing to fetch the desired timeframe data, combining the Disinformation data with the Twitter Archives data (Non-Disinformation) using distinct screen names indexing to avoid duplication. The above queries are available in the “C\_R\_T.sqbpro” file on GitHub. The python code for combining the data for all three countries can be found in **“C\_R\_T\_Concatination.py” on GitHub. The Non-Disinformation data (Others) from June 2019 are concatenated using the “Others\_Concatination.py” file on GitHub. The language distributions code for each of Turkey, and Russia can be found in “Russia\_Lang\_distribution.ipynb” and “Turkey\_Lang\_Distribution.ipynb” respectively. The code for Classification model development for training, and testing for each of Russia, and Turkey can be found in “June\_Russia.ipynb” and “June\_Turkey.ipynb” respectively.**

**We hosted our files using a “GNU General Public License v3.0”. The “GNU General Public License v3.0” is a free, copyleft license for software and other kinds of works which gives the permissions for**Commercial Use, Modification, Distribution, Patent use, and Private use. It also has these limitations: Liability, and Warranty.

# Appendix : Risk Section

| Risk Name | Description | Probability | Impact | Mitigation |
| --- | --- | --- | --- | --- |
| Accuracy | As most of the tweets are in English, it might be difficult to find a pattern in data for the model | high | high | Not sure how to tackle it but to get the data in which tweets must be other than English |
| Useful Data | Since most of the tweets are in English it might be difficult to identify a pattern in data for the classification model | high | high | Tweets that are in the respective class must be obtained and other languages tweets must be handled using necessary measures. |
| Combining Datasets | The data from different sources are in different formats and is required to be transformed without losing data. | Moderate | Moderate | Transforming is going on in Python using inbuilt functions and necessary trial and error scenarios are being used. |
| Preprocessing and reducing noise. | Since most of the tweets are in English it might be difficult to identify a pattern in data for the classification model. | high | high | Tweets that are in the respective class must be obtained and other languages tweets must be handled using necessary measures. |
| Time Consuming | Data is very large, and analysis takes a lot of time | High | High | We use the Amazon Web Services to minimize the time. |
| Deep Leaning Models | We never worked on Deep Learning Models and learning them yet. | Moderate | Moderate | We are coping up by splitting up the work among the developers. |
| Complex Functionality | The architecture is complex and since the data is huge, it gets difficult for trial and error to be performed on the data using this model. | High | High | We can modify the batch sizes that is available in the algorithm hyperparameters. |

Table : Risk table

# Appendix : Agile Development Workflow

9 Dec 2020

Sprint 5 – 30 Nov 2020

Sprint 4 – 16 Nov 2020

Sprint 3 – 26 Oct 2020

Week1 – 24 Aug 2020

Sprint – 31Aug 2020

Sprint 1 – 14 Sep 2020

Sprint 2 – 5 Oct 2020

# Appendix : References

|  |  |
| --- | --- |
| [1] | E. S. A. J. GOTTFRIED, "News Use Across Social Media Platforms 2017," Pew Research Center, 7 September 2017. [Online]. Available: https://www.journalism.org/2017/09/07/news-use-across-social-media-platforms-2017/. |
| [2] | N. R. Chris McCormick, 22 July 2019. [Online]. Available: https://mccormickml.com/2019/07/22/BERT-fine-tuning/. |
| [3] | R. Munro, "youtube," 26 August 2020. [Online]. Available: https://www.youtube.com/watch?v=cwGVbEWoM7U&ab\_channel=RobertMunro. |
| [4] | M. S. Z. Rizvi, "AnalyticsVidhya," 25 September 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/. |
| [5] | P. Joshi, "AnalyticsVidhya," 19 June 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/?utm\_source=blog&utm\_medium=demystifying-bert-groundbreaking-nlp-framework. |
| [6] | R. Horev, "towardsdatascience," 10 November 2018. [Online]. Available: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270#:~:text=How%20BERT%20works,a%20prediction%20for%20the%20task.. |
| [7] | P. SRIVASTAVA, "Essentials of Deep Learning : Introduction to Long Short Term Memory," Analytics Vidhya, 10 December 2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/. |
| [8] | S. RAY, "Understanding Support Vector Machine(SVM) algorithm," Analytics Vidhya, 13 September 2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/. |
| [9] | V. Valkov, "youtube," 19 April 2020. [Online]. Available: https://www.youtube.com/watch?v=8N-nM3QW7O0&ab\_channel=VenelinValkov. |
| [10] | K. Naik, "youtube," 29 August 2020. [Online]. Available: https://www.youtube.com/watch?v=cwGVbEWoM7U&ab\_channel=RobertMunro. |
| [11] | V. Valkov, "youtube," 11 April 2020. [Online]. Available: https://www.youtube.com/watch?v=Osj0Z6rwJB4&ab\_channel=VenelinValkov. |
|  |  |