Spot a Twitter Bot

A cloud pipeline

Lieby Cardoso
746103

Submission date: ***12/29/2019***

Contents

Abstract

Since the 2016 US elections, the possibility of interference of twitter automates accounts (bots) meddling facts with fake news, spreading the content through an intensely social activity has gained the spotlight. As the Canadian Federal election approached we collected a large number of messages from official Canadians candidates twitter account and from random accounts that used specifics hashtags. A sentiment analysis algorithm was applied to the tweet content and the user description text. Others feature as the number of followers and days since user signup were taken into consideration. Data analysis techniques used to analyze and summarize the behavior of well-known bots and not bots accounts and the result shows differences between both accounts types. The differences draw in a dashboard provide pieces of evidence that a supervised algorithm can help classify potential bots users and identify their social interaction behavior.

Introduction

Social media platforms like Facebook, Instagram and Twitter have become a popular business option for target and spread information about products, services, and companies' brands.

Among then Twitter has a couple of significant features that allow people to engage with brands, messages, and trending topics. In a recent earnings report released, Twitter counts with an average of 321 million users, which 126 million uses the platform daily to find faster news and events, locally and globally as well. This massive number it is only obfuscated by Facebook's 1.2 billion daily users.

Companies learned how to take advantage of these large numbers of potential consumers, brands, and anyone with an interest in propagating a message that can utilize Twitter to reach and connect with their group of spectators. The platform becomes a channel for sharing a large variety of information including false ones. Hashtags give to people and companies an opportunity to target and take an interest in discussions about themes relevant to you or their business. Additionally, the channel can be used to reach out to new target users who may be interested in the services offered, leading to the exploitation of people who trust the message delivered.

Jack Dorsey, Twitter's CEO, said the company's strategy is centered on four broad objectives: promoting healthy conversation on the platform, making it easier for users to participate in conversations, improving the platform for advertisers and focusing on new technology to improve the service.

The Pew Research Center, an organization that defined themselves as “a nonpartisan fact tank that informs the public about the issues, attitudes, and trends shaping the world” and are responsible for executing public opinion polling, demographic research, media content analysis conducted a survey of 2,791 U.S. adult Twitter user. Their finds show that the median user age is 40, 42% have a college degree, 41% report an annual income above the $75,000 and most important, 80% of the content is produced by 10% of most active users and that 10% usually tweety about politics related themes.

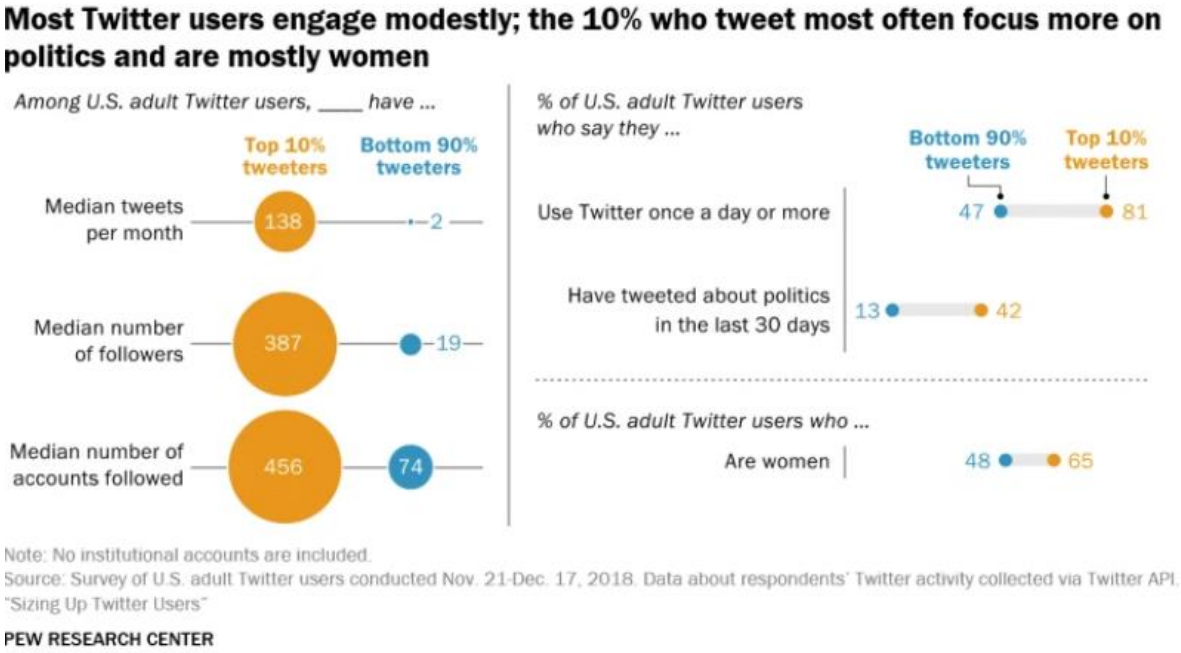

Figure 1: User Engagement. Reprinted from Sizing Up Twitter Users by PEW Research Center, April 24, 2019, retrieved from https://www.pewresearch.org/internet/2019/04/24/sizing-up-twitter-users/

In this context where the top 10% twitter users ruled the vast majority of the messages produced, consumers are ingesting all types of information, sometimes without any filter or critical assessment of origin.

As demonstrated by the PEW research center, tweets about politics are commonplace. Political candidates are not far behind, they have shown intense activity on twitter, where they use to spread their political platform and ideas among target voters.

In this paper, we take a look into 3 most prominent Canadian politicians accounts being them:
Justin Trudeau the Prime Minister and representative of the Liberal Party, Jagmeet Singh Leader of Canada's New Democratic Party and Andrew Scheer Leader of Canada’s Conservatives and Leader of the Official Opposition.

Our work focuses on the design and development of a cloud-based solution capable of gathering, ingest and support a streaming twitter structure, further store the data for consumption and delivery of insight on the streamed data.

At first, 189,382 tweets were collected in a 15 days period; a consumer-subscriber application deployed on Google Cloud Kubernetes is activated by a Pubsub message system and streamed to Bigquery.

Secondly, a sentiment related to the tweet text was evaluated, being the classification group positive, neutral or negative based on the polarity score. The BlobText Library was used to find out the sentiment associated with each text. A method for cleaning and deleting non-alphabetic characters was also performed to prepare the text. The same logic was applied to the user description text.

Thirdly, after the data collection, cleansing, and performance of sentiment analysis, this paper describes the collection of data to create a training dataset with automated (bots), and humans account information. Several machine learning classification algorithms were applied and evaluated to determine if a user i is a Bot or a human, based on his account profile. Metrics, score, and model deployment are also described.

A conclusion is provided, and a future study proposed once, at this point, we are able to classify if a user is a bot or not based on his profile, leaving for another moment the classification based on the tweet content.

Methods

In order to be able to create a dataset with user and account information, it was first necessary to design a cloud service ecosystem capable of supporting data streaming operations and aligned with the needs of this type of service and social media data collection characteristics.

The Google cloud platform (GCP) was chosen for offering $ 300 bonus credit, a 1-year free tier and contain all the services needed to support the product. In order to verify the platform capacity and functionalities, a proof of concept (POC) project was developed following Google Github directory tutorial example using Google Kubernetes Engine (GKE) that builds a pipeline to stream data into BigQuery, all on the shoulders of PubSub. A public docker container image (gcr.io/google-samples/pubsub-bq-pipe:v5) hosted on Google container registry runs inside Kubernetes pods.

After a successful deployment, a batch with 8.000 tweets were streamed from twitter to a BigQuery table. The subscriber application created a fully functional dataset with all the tweets features.

As the POC project handling and delivery the streaming of data according to our expectations, we decided to build a similar project that in the end result in a full dataset that allows the extraction of insights.

The final GCP solution built is exhibited in fig 7. All the steps implemented to develop the pipeline from collect the data to the visualization are described in the following topics.

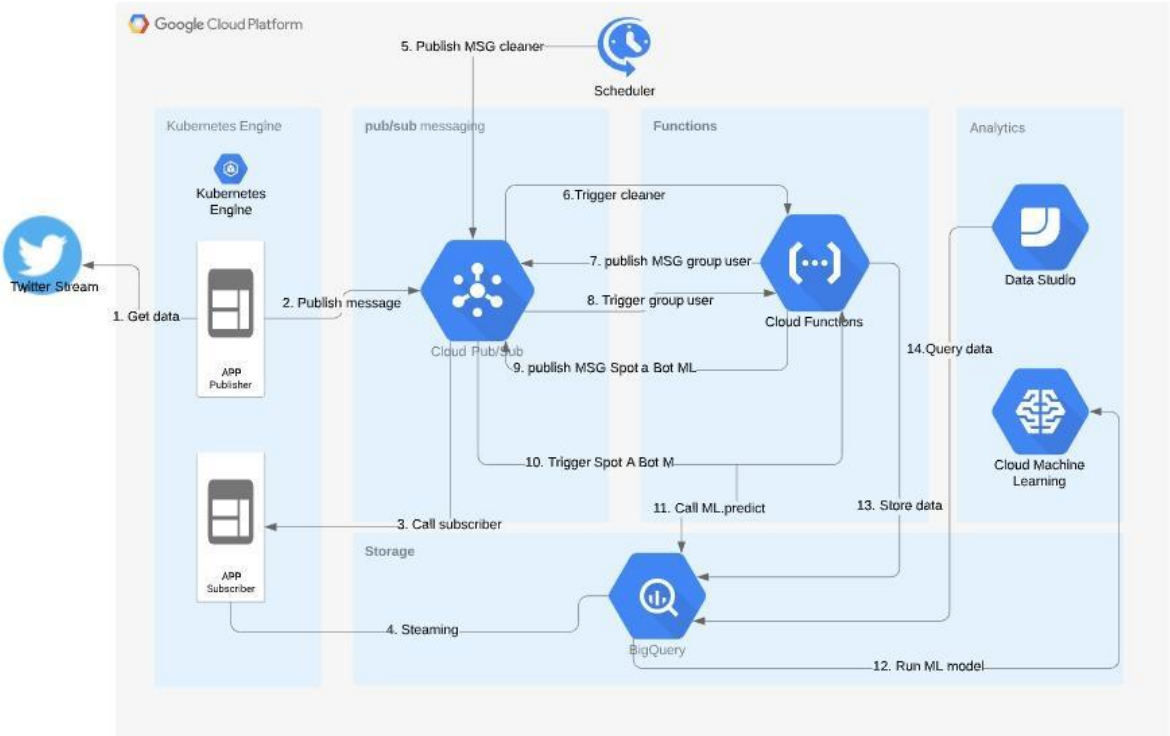

Fig 7: Capstone project structure

1. Get Data

All information processed by this project is provided by Twitter APIs that return a list of key-value pairs of data encoded using JavaScript Object Notation (JSON). The object returned has named attributes and associated values. As the Twitter developer website mention “Each Tweet has an author, a message, a unique ID, a timestamp of when it was posted, and sometimes geo metadata shared by the user. Each user has a Twitter name, an ID, a number of followers, and most often an account bio. With each Tweet, we also generate "entity" objects, which are arrays of common Tweet contents such as hashtags, mentions, media, and links. If there are links, the JSON payload can also provide metadata such as the fully unwound URL and the webpage’s title and description. “

Which object retrieved can have over 150 attributes associated with it. To accomplish this project objective we choose to work with 16 attributes related to sentiment analysis.

Attributes chose:

id	Int64	A sequence of numbers that uniquely identifies a user
screen_name	String	Alias defined by the user
location	String	Location defined by the user
description	String	String describing the account
verified	Boolean	True for verified accounts
followers_count	Int	The number of followers

friends_count	Int	Number of accounts followed
listed_count	Int	Number of a list the user is part of.
favourites_count	Int	The number of liked tweets
statuses_count	Int	The number of Tweets
created_at	String	Datetime that the account was created
text	String	Tweet text
quote_count	Integer	Number of times a tweed was quoted
reply_count	Int	Number of times a tweed was replied to
retweet_count	Int	Number of times a Tweet was retweeted
favorite_count	Integer	Number of times a Tweet was liked

Table 1: Tweets attributes

2. Publish message

Google PubSub is a messaging system that allows secure communication between applications. In order to use this structure, an application was developed in python 3 with the purpose of calling the Twitter API, filtering the attributes mentioned in table 1 that would be used. After the filtering action, the message is ready to be codified into base64 and published to the PubSub topic “tweets”.

Before to use the Twitter API it is necessary to create a developer account on Twitter, create an app resource as all. Once the app is available, Twitter provides a pair of token and key, used by the application to certify to Twitter that the application is a secure one.

Python has libraries that support automated actions on Twitter as gather data and post a tweet. As Twitter recently allows tweets to exceed 140 characters, we decide to use Tweepy to interact with Twitter.

An operation system variable named TWSTREAMMODE is set to inform the application in which two types of calls should be performed. When TWSTREAMMODE is equal to “timeline” the application calls tweepy.Cursor method passing as parameter the politician user_name. If TWSTREAMMODE is different from “timeline” a stream connection is established through the method stream.filter and a list of hashtags is tracked. To access de tweet data we override the Tweepy on_data method.

Hashtags tracked: ['#cdnpoli', '#elxn43','#CanadaElection2019', '#canpoli', '#CanadianElection', '#JustinTrudeau', '#jagmeetsingh', '#AndrewScheer', 'CPC_HQ', 'liberal_party', '#ChooseForward', '#ndp', 'InItForYou', '#cpc', '#lpc']

3. Call Subscriber

To access the message published into the topic by the publisher’s application it is essential to create a subscription to the topic. PubSub sends the message to a subscriber, then the subscriber acknowledges the message. The subscriber application pubsub-bq.py creates a channel communication with PubSub through the client and listens to messages posted on the topic. The communication is established through the callback

argument implemented in the subscribe method of the pubsub_v1 object, making easier for the subscriber to pull messages from the topic.

During the callback, the message is intercepted, translated form base64 and sent to a Bigquery table named tweet.

4. Streaming

In this project context, we call streaming a continuous transfer of data between Twitter's API and BigQuery. To keep the flow of information automated running the publisher and subscriber application were containerized into a docker container and deployed into Kubernetes. One master node with 3 pods was created, one pod was assigned to the publisher and 2 pods for the subscriber.

The Docker container is created by Google Cloud Build and published to the google container registry service. Using YAML configuration files that contain the container address, both applications were deployed to the cloud. Once the deployment is successfully done, Kubernetes orchestrate wich app pods read data, publish it and which subscribes to the PubSub topic.

5. Publish MSG cleaner

Every night at 11:00 Cloud Scheduler publish a generic message to clean-bq-tweet PubSub topic. The main goal is to trigger cf-clean-bq-tweet cloud function.

6. Trigger Cleaner

Cf-clean-bq-tweet is a cloud function responsible mainly for clean the data stored in BigQuery tweet table. To remove wanted characters and apply the sentiment analysis the steps are followed:

1. Fetches data from the tweet table and stores it into a pandas data frame
2. Add columns:
    - Hashtags: Array of hashtags extracted from tweet text
    - hashtag_count: Number of hashtags extracted
    - lang: Detects the tweet language
    - creation_days: Number of days between creation_at and processing date
3. Drop duplicated lines
4. Drop Tweets not in English (lang != “en”)
5. Clean text and description
    - Remove mentions (@username)
    - Remove special characters
    - Remove Links
    - Remove “rt”
6. Applies Sentiment Analysis on both features text and description
    - Tokenize

- Remove stops words
- Lemmatization
- Creates a Textblob object
- Adds the Textblob sentiment score to polarity and subjectivity columns
- Based on the sentiment score assign “positive”, “neutral” or “negative” to the sentiment column

7. Store the data into sentiment table
8. Publish a message to aggregate-user PubSub topic

hashtags	hashtags_count	polarity	subjectivity	sentiment	creation_days	description_coun	desc_polarity	desc_subjectivity	desc_sentiment
['cbcfakenews', 'c	5	0	0	neutral	3	1	0	0	neutral
['cdnpoli', 'onpoli'	7	0	0	neutral	3	1	0	0	neutral
['InformedConse	9	0	0	neutral	3	7	0	0.1	neutral
['PPC', 'immigran	4	0.3	0.5	positive	3	10	0	0	neutral
['ableg', 'cdnpoli']	2	0.2	0	positive	3	10	0	0	neutral
['cpc', 'leadership	4	0	0.4	neutral	3	14	0	0.3	neutral

Table 2: Sentiment table

7. Publish MSG group user

When the cloud function cf-clean-bq-tweet process successfully, a message is sent to aggregate-user PubSub topic. This event triggers cf-aggregate-user cloud function.

8. Trigger group user

cf-aggregate-user is a cloud function called after the tweet is cleaned and sentiment analysis is applied. Its purpose is aggregate sentiment table by username, and prepare the data for logistic regression model consumption.

Part of the preparation is to turn the sentiment column into your dummy version: sentiment_negative, sentiment_neutral, and sentiment_positive. When it's done, the data frame data is inserted into the Bigquery user table.

9. Publish MSG Spot a Bot ML

When the cloud function cf-aggregate-user process successfully, a message is sent to model-logistic-reg PubSub topic. This event triggers cf-model-logistic cloud function.

10. Trigger Spot a Bot ML

Once the data is ready to consumption cf-model-logistic cloud function is executed. Its function is to execute the logistic regression model to classify whether the user is a robot or not. The model is deployed inside BigQuery and is executed by calling the Select ML.predict (dataset.model_name) command.

11. Call ML predict

ML.predict SQL statement executed through cf-model-logist.

12.Run ML Model

A logistic regression model with L1 regularization = 0.10 predicts outcomes based on the input table user and return as many rows as the user table plus predict label column.

13.Store Data

All data collected, processed and inputted into the model are stored into BigQuery.
Tables:

Table Name	Description
Tweet	Raw twitter data
Sentiment	Cleaned twitter data + sentiment analysis
User	Sentiment table data aggregated by user_name + sentiment dummies
Predicted	User table data + predicted label indicating if a user is a bot or not

Table 3: BigQuery tables

14. Query Data - Data studio

All processing was done in GCP, so the most natural option was to choose Data Studio as a data visualization tool. The integration with BigQuery datasets and the intuitive interface made it easy to create dashboards.

To display the data processed 3 dashboards were created.

- **Dashboard 1: Overview**

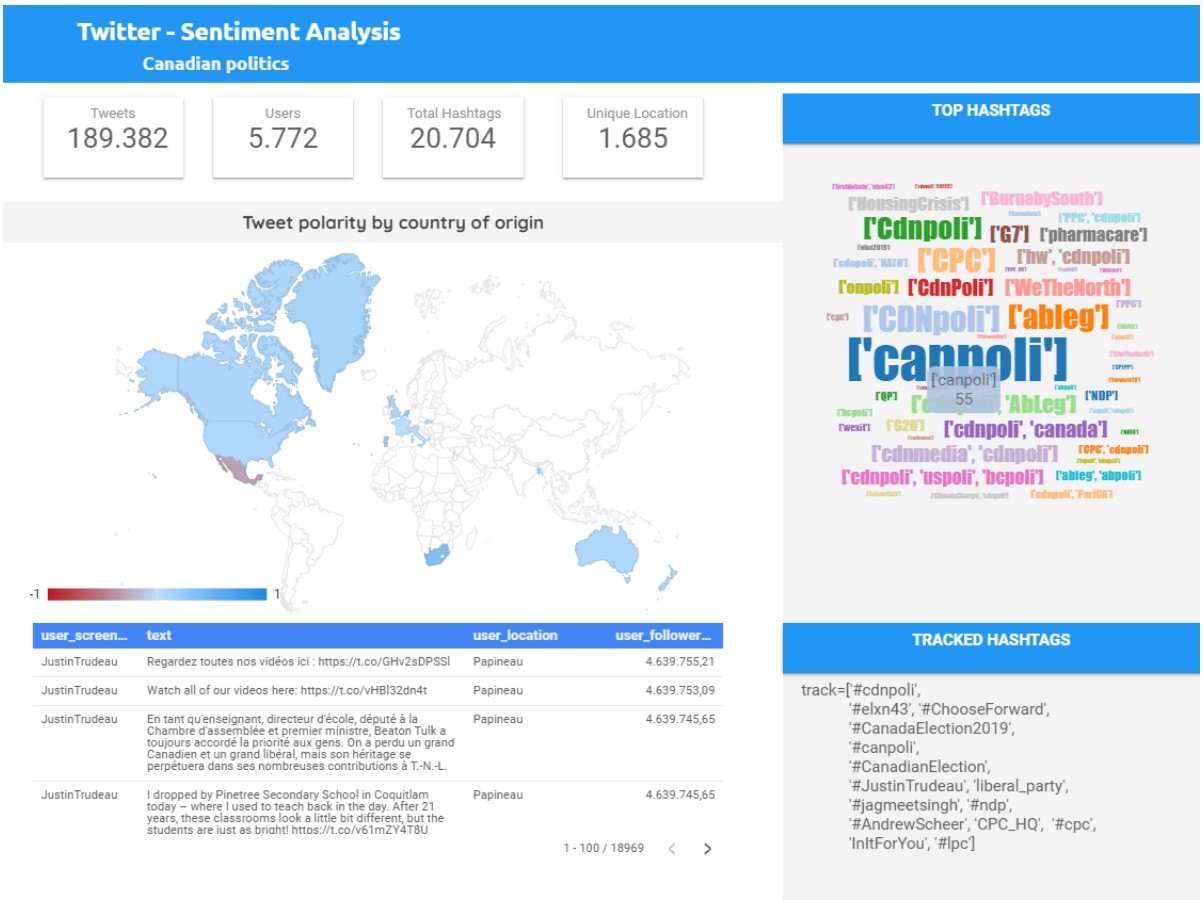

Fig 8: Dashboard 1 - Overview

- **Dashboard 2: Sentiment Analysis**

Focus on sentiment related with Canadian politics tweets. Justin Trudeau, Jagmeet Singh and Andrew Scheer tweets displayed in a timeline graph, groped by sentiment, besides the sample of messages and sentiment score.

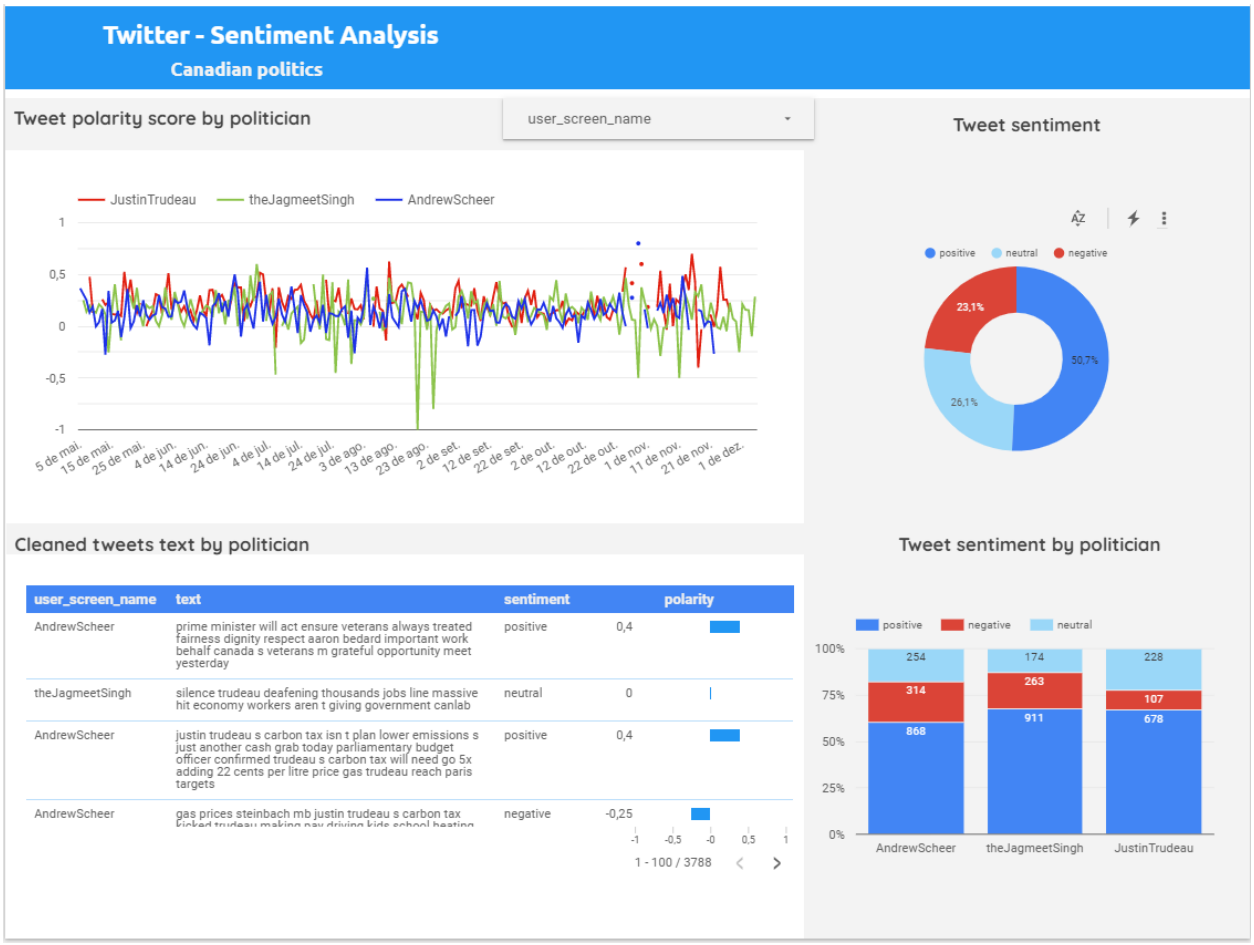

Fig 9: Dashboard 2 - Sentiment analysis

- Dashboard 3: Spot A Bot

Dashboard 3 is responsible for displaying the predicted data based on the users profile.

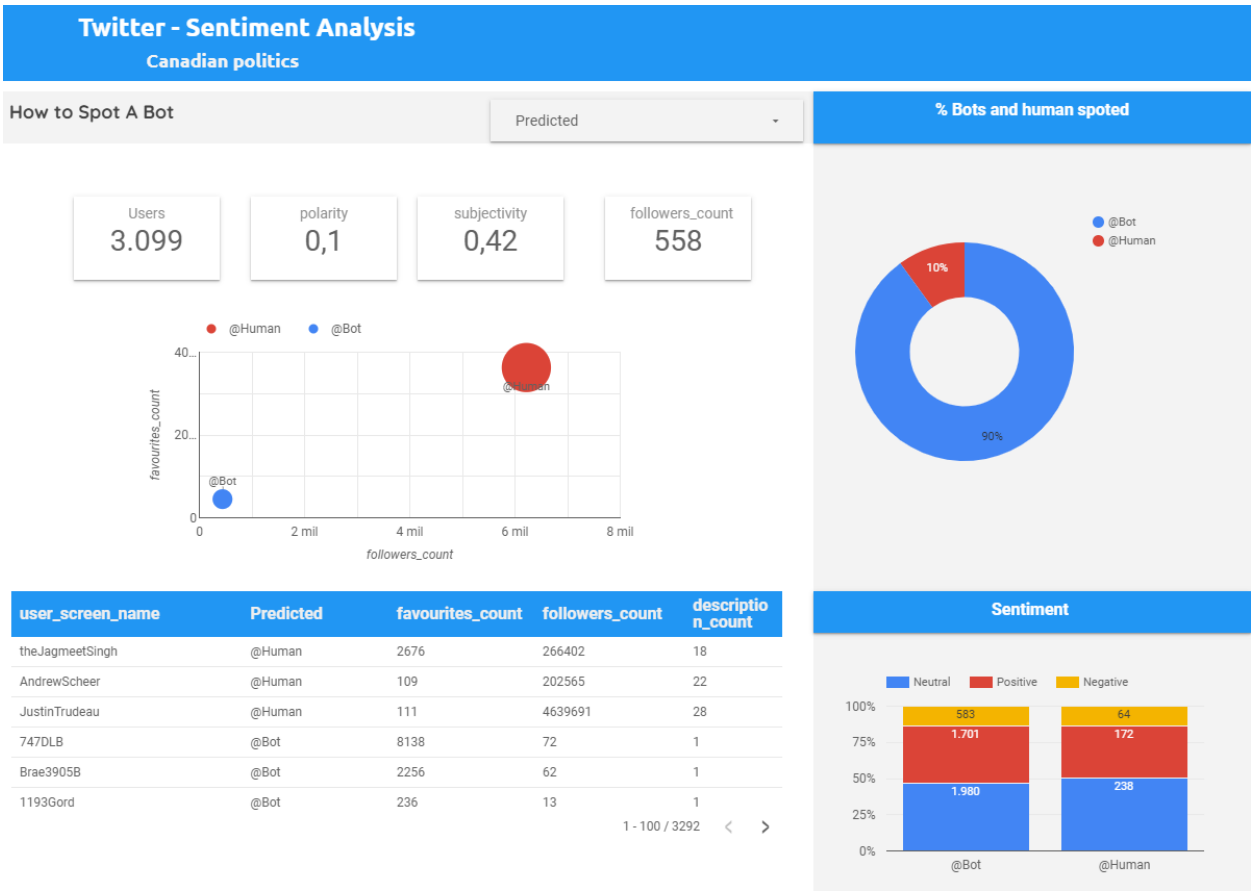

Fig 10: Dashboard 3 - Spot A Bot

Spot a Bot

Data Collection - Train DataSet

One of the biggest challenges of this project was getting enough information to compose a dataset to train the model. In the training dataset, you need to have information about Twitter users who are admittedly robust, human user profiles and verified users who are usually recognized Twitter personalities.

An obstacle in building the dataset is because when Twitter detects that an account has malicious activity, the account is suspended and all content is removed from public access. On January 31, 2018, Twitter publishes a review of the 2016 US election :

"With our current capabilities, we detect and block approximately 523,000 suspicious logins daily for being generated through automation. In December 2017, our systems identified and challenged more than 6.4 million suspicious accounts globally per week— a 60% increase in our detection rate from October 2017.
Alongside these improvements, we're continuing to expand enforcement of our developer and automation rules. Since June 2017, we've removed more than 220,000 applications in violation of our rules, collectively responsible for more than 2.2 billion low-quality Tweets. "

Thankfully NBC News made it public a dataset with a list of blocked profiles that became part of the data source along with Andrew Caide's project completed for Harvard's CS109 Data Science, 2018.

Applied SMOTE - Synthetic Minority Over-sampling Technique to balance the minority sample of Bot accounts data.

Data Analysis

To prepare the training dataset to be consumed by a classification model, the sentiment variable was transformed into three dummies: sentiment_negative, sentiment_neutral, and sentiment_positive. After the transformation, perform a label encoder was not necessary once none of the features are categorical.

A Pearson pairwise correlation of columns demonstrates a strong correlation (> |0.6|) between Bot and followers_count, listed_count, statuses_count, verified and creation_days.

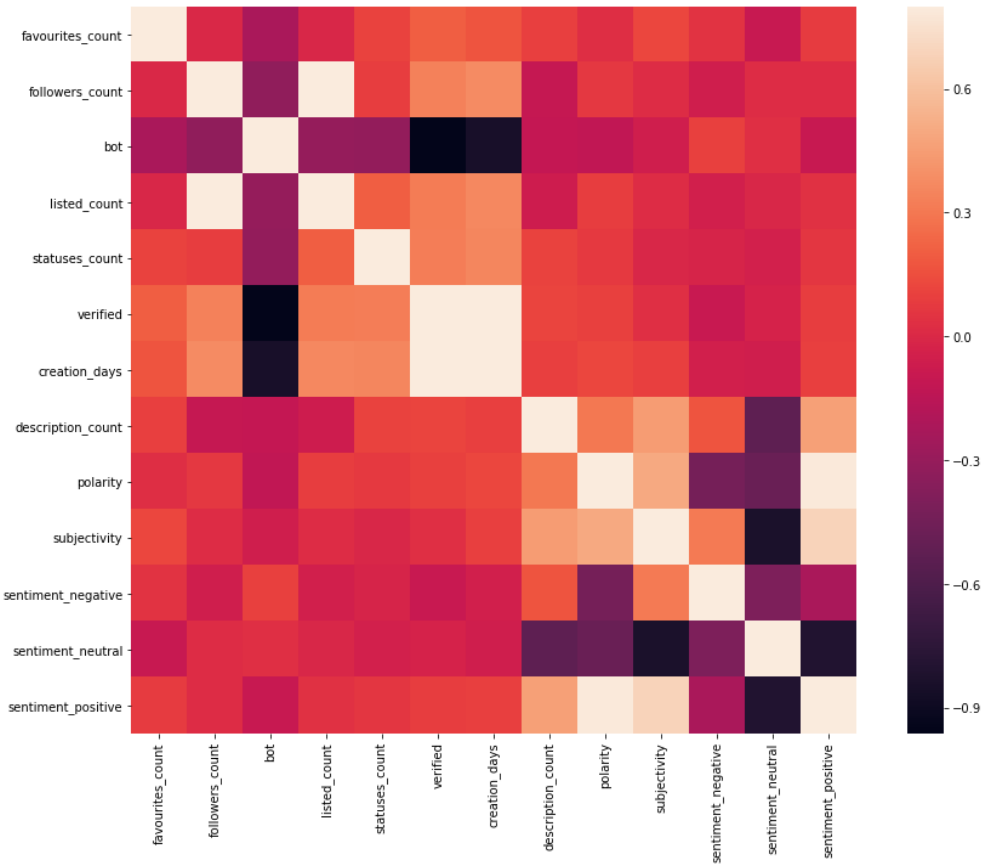

Fig 11: Pearson Correlation plot

Pair plot of features, aggregated by bot.

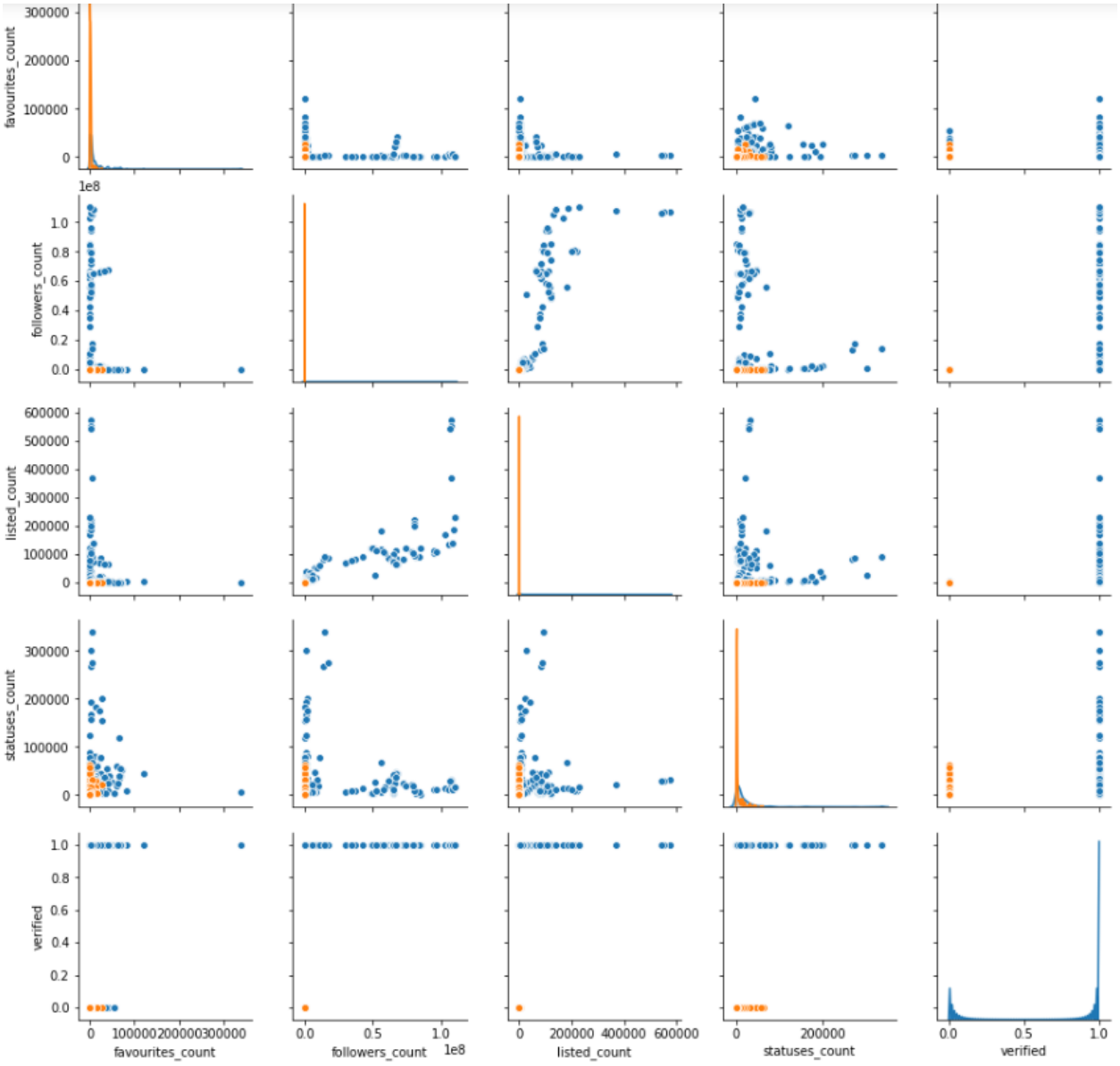

Fig 12: pairplot 1

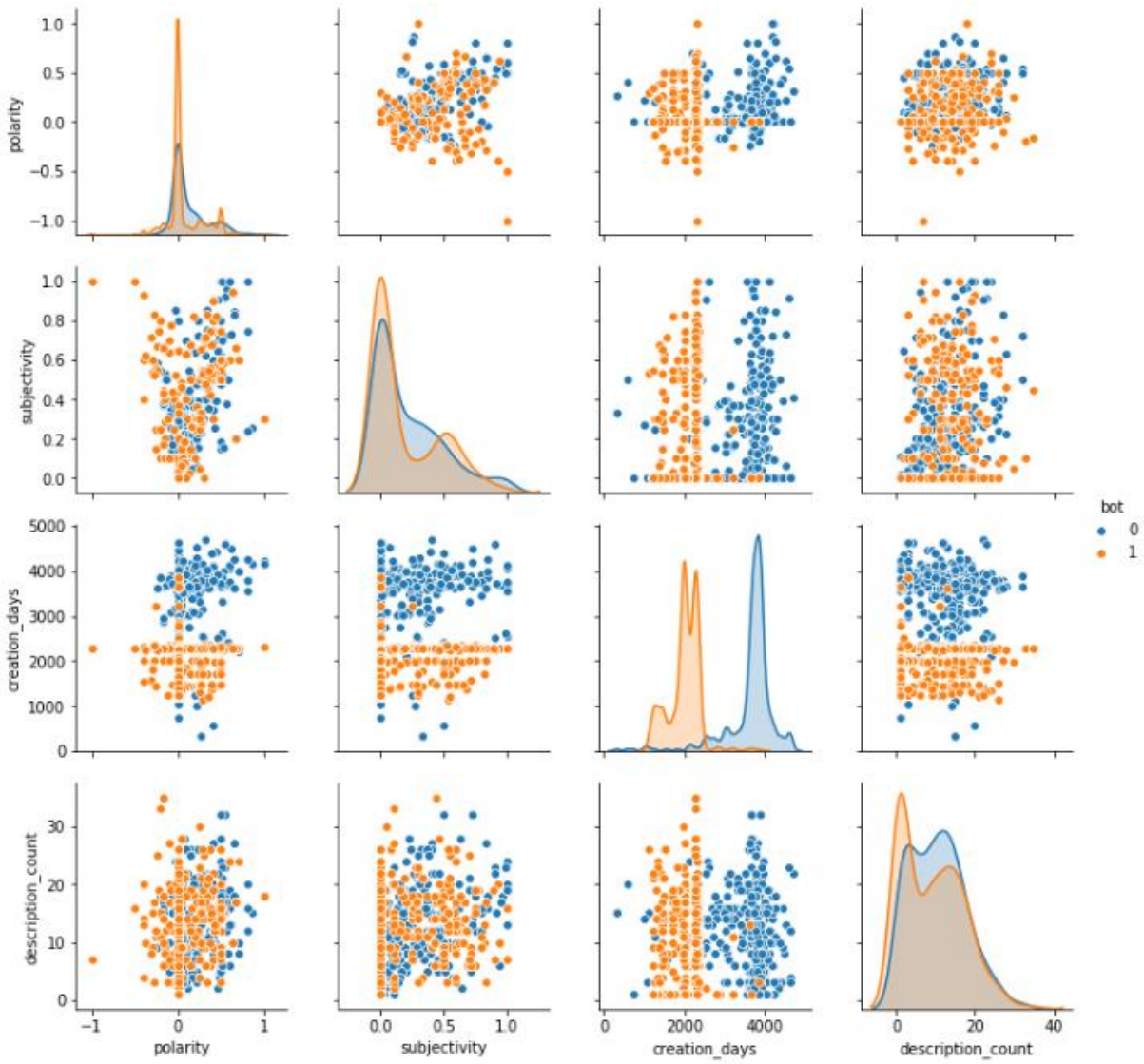

Fig 13: pairplot 2

Pairs of plot visualization give us valuable insights. By visualizing the relationship between pairs of variables, it is possible to notice a difference in the behavior of bots and non-bots accounts. Bots (1, orange) has a smaller number of followers, their tweets are less liked, and visibly their accounts are less than 1 year old. Through the graphics, we can also confirm that the authenticated accounts do not belong to bots, as we expected.

Model Selection

Before deciding to go with Bigquery's ML.predict, some classifier models were evaluated. Below is a table with the model and performance obtained.

Model	Precision	Recall	F1	Accuracy	Parameters
LogisticRegression	0.97	0.92	0.94	0.96	C=0.5, class_weight=None, dual=False,

					fit_intercept=True, intercept_scaling=1, l1_ratio=None, max_iter=100, multi_class='auto', n_jobs=None, penalty='l2', random_state=42, solver='liblinear', tol=0.0001, verbose=0, warm_start=False
GradientBoostingClassifier	0.96	0.95	0.96	0.96	ccp_alpha=0.0,criterion='friedman_mse', init=None, learning_rate=1.0, loss='deviance', max_depth=3, max_features=0.5, max_leaf_nodes=100, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=50, n_iter_no_change=None, presort='deprecated', random_state=42, subsample=1.0, tol=0.0001, validation_fraction=0.1, verbose=0, warm_start=False
RandomForestClassifier	0.97	0.96	0.96	0.97	bootstrap=True, class_weight=None, criterion='gini', max_depth=5, max_features=2, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=3, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1, oob_score=False, random_state=None, verbose=0, warm_start=False

Table 4: Model performance

The final model chosen was Logistic Regression deployed on Bigquery. Modeled performed well with training data as well as test data. BigQuery itself applies split techniques, split the dataset, and shows the model evaluation results as shown below.

Model details

Model ID	heroic-gamma-254018:capstonettw.spot_bot
Date created	Nov 23, 2019, 11:47:35 PM
Model expiration	Never
Date modified	Nov 23, 2019, 11:47:35 PM
Data location	US
Model type	LOGISTIC_REGRESSION
Loss type	Mean log loss

Training options

Max allowed iterations	20
Actual iterations	20
L1 regularization	0.10
L2 regularization	0.00
Early stop	true
Min relative progress	0.01
Learn rate strategy	Line search
Line search initial learn rate	0.10

Aggregate metrics

Log loss	0.0163
ROC AUC	1.0020

Score threshold

Positive class threshold 0.0037

Positive class	1
Negative class	0
Precision	0.9560
Recall	1.0000
Accuracy	0.9685
F1 score	0.9775

Use this slider above to see which score threshold works best for your model.

Confusion matrix

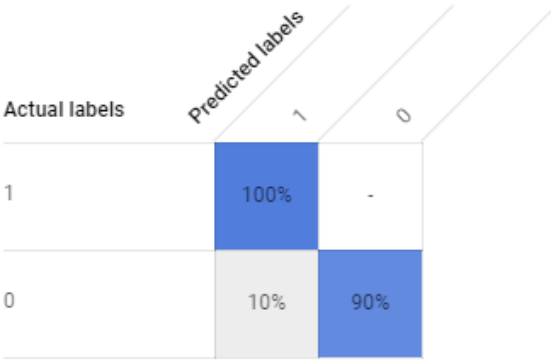

Fig 14: Logistic Regression evaluation

Continuous deployment

This project complies with agile methodology and best practices for software delivery automation. Project source codes are controlled by the git version source controller. The GitHub repository stores these files and does versioning. The repository is synchronized with the Google Cloud Build service, every time a change is made, a trigger is running, which automates docker container generation. Once the container is ready, Google Cloud Build published the container to Google container-registry.

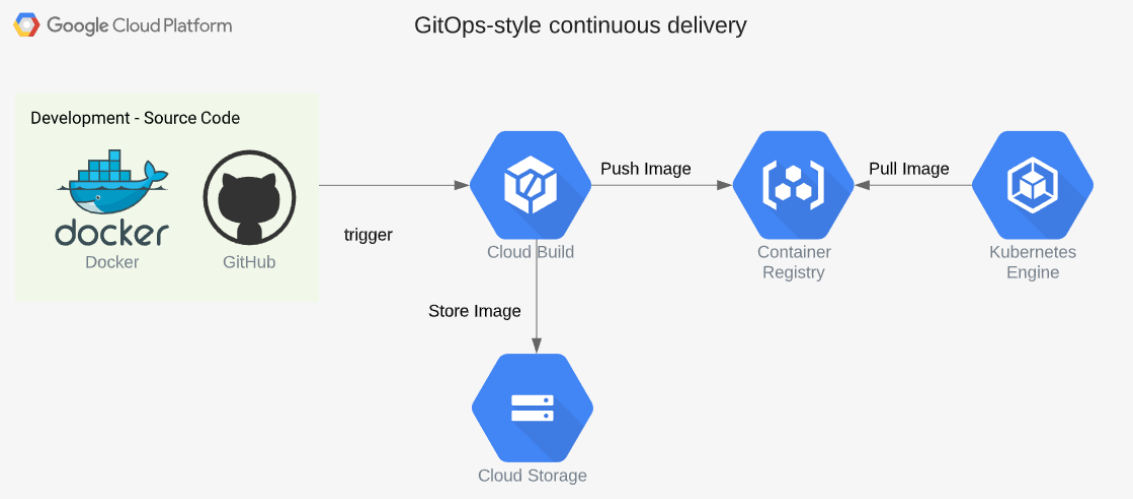

Fig 15: Continuous deployment

Problems encountered

As the projects aim to provide a platform for streaming and data analytics, as expected, some problems were encountered from the execution phase to the presentation of data.

From the beginning, in the requirements analysis phase, It was difficult to reach an agreement regarding the scope of the project, as the team members have different knowledge background and expectations, was hard to design a project idea that would take advantage of the diversity of knowledge of the members while allowing the use in the real world of the knowledge gained in the course. By common agreement, the solution chosen is made up of 50% cloud service development and usage and 50% analytics. This was the most important decision that impacted all the development of the project. As there was a lack of experience in implementing cloud solutions and lacking knowledge of some GCP tools, more than 80% of the project effort was focused on what would have been just 50% of cloud service development and usage defined before.

Due to the restriction of build docker containers in the windows home version, was necessary to find a new way to generate and publish Docker containers. The final decision was to use Google Cloud Build to build the Docker container and publish it to Google Container Registry. Which in general was a good decision because it facilitated the creation of continuous project deployment.

Once we were sure that data streaming would be done in python and packaged in a docker container, it was difficult to understand how the Kubernetes environment worked and in the early errors, it took a long time to troubleshoot, interpret the logs and provide a solution. As an example, at some point a failed deployment returned this message “Does not have minimum availability”, we spent hours trying to understand it was a configuration or a technical problem in the node. It was actually a base64 encode problem in the application's Python code, which blocked the program from running successfully.

Creating a dataset with account information that is verified as verified and unverified robots and human accounts was an almost impossible mission. Even though the resulting dataset does not have the required

amount of users to apply the classification algorithm, we decided to proceed with the project in that direction, leaving for future improvement the insertion of more records on this dataset.

Other minor problems were also faced as cleaning cloud function using more memory than the defined leading to the function crash. To solve, I reduced the number of tasks performed by the function, which in the end is the purpose of serverless functions.

Another small error was the word cloud graphic that was developed by the Data Studio community. Sometimes the application fails to render the data, but as the value-added was higher, we left the graph in the dashboard.

Results

The full pipeline benefited from PubSub's messaging system, and it was on his shoulders that the entire structure was based. PubSub provided for this project a reliable and fast way of exchanging messages between applications that aren't hosted in the same service. We ended up having 5 topics:

Name	Purpose
tweets	Store tweets messages for subscriber consumption
clean-bq-tweet	Trigger a cloud function that Cleans and Preprocess tweets content
aggregate-user	Trigger a cloud function that group data by user
model-logistic-reg	Trigger a cloud function that runs ML.predict
cloud-builds	Trigger container builder

Table 5: PubSub Topics List

The Kubernetes node and 3 pods were perfectly orchestrated delivering what is promised by Google, as a solution developer we only had to worry about coding and requirements, let in for Google to manage the state and balance of virtual machines.

Kubernetes cluster created runs the kubernetes 1.13.11-gke.14 version, with 3 nodes and us-central1-a as Master/Node zone. The default maximum pods per node are equal to 110.

Name	Status	CPU requested	CPU allocatable	Memory requested	Memory allocatable	Storage requested	Storage allocatable
node1	Ready	461 mCPU	940 mCPU	346.03 MB	2.77 GB	0 B	0 B
node 2	Ready	249 mCPU	940 mCPU	340.79 MB	2.77 GB	0 B	0 B
node3	Ready	541 mCPU	940 mCPU	472.68 MB	2.77 GB	0 B	0 B

Table 6: Kubernetes nodes List

To run the publisher and subscriber application the configuration used is adequate.

Cloud function's serverless functions have made it much easier to run small units of programs in the cloud and because of them, we don't need to develop any complete data cleansing/data aggregation applications.

Function Name	Region	Trigger	Runtime	Memory allocated
cf-aggregate-user	us-central1	Topic: aggregate-user	Python 3.7	256 MB
cf-clean-bq-tweet	us-central1	Topic: clean-bq-tweet	Python 3.7	256 MB
cf-model-logistic	us-central1	Topic: model-logistic-reg	Python 3.7	256 MB

Table 7: Cloud Functions List

BigQuery plays an important role in the project because, in addition to being the common data repository, we use python libraries that integrate well with python and allow us to use standard SQL to query and store data directly in a pandas data frame. In addition, the ML.predict function allowed us to create a Logistic Regression model, that can be called through a query every time a prediction needs to be made. We executed a simple insert + select SQL statement to classify if a Twitter user is a bot or not, took 1.5 minutes to select, classify and insert 4.500 records.

Dataset summary:

Number tables	Number rows	size Megas
10	302,783	137

Table 8: Dataset capstonettw summary

The Spot a bot model is a logistic regression model with L1 equal to 0.1 and loss type equal Mean log loss. Train and model evaluation is provided by Google BigQuery using the project training table inputted.

Model	Recall	Precision	Accuracy	F1 score	Log Loss	ROC
LOGISTIC REGRESSION	1.00	0.95	0.96	0.97	0.0163	1.002

Table 9: Model score

As we kept the data visualization design simple with no complex graphs, Data Studio had all the necessary elements to produce 3 fully functional dashboards. With easy integration with BigQuery, an intuitive interface, a shareable public link address, Data Studio was a good decision for the project, but in case of a project growth, another more robust visualization tool will be needed.

In the table below you can see the total calls that the project made to each service API. 41.318 request to PubSub, 11.618 to BigQuery and 942 to Cloud Functions.

Name	Requests	Errors (%)	Latency, median (ms)	Latency, 95% (ms)
Compute Engine API	426,088	0	219	484
Stackdriver Monitoring API	418,425	0	45	258
Stackdriver Logging API	349,329	0	91	130
Cloud Pub/Sub API	41,318	3	8	23,303
BigQuery API	11,618	0		
Cloud Functions API	942	0	204	466
Cloud Scheduler API	131	3	45	123

Fig 16: API usage

Besides the design and implementation of the cloud solution, part of the work aimed to analyze the pattern of sentiment involved in the Canadian election period. Analysis of over 189,000 tweets and 5,000 users suggests that from May to November 2019 Andrew Scheers was the one with the most negative speech from the three candidates, in contrast, Justin Trudeau delivered a more moderate speech with more positive tweets and neutral. Interestingly, we also noticed that many tweets with negative to neutral polarity were generated by users located in other countries of the world, such as Mexico with a polarity of -0.31 and negative sentiment.

Regarding Spot A Bot we have evaluated the rating of 5,084 users. Bots have a tendency to have fewer followers and fewer liked tweets. Surprisingly the average percentage polarity of sentiment expressed between bots and non-bots is similar. The logistic regression model was capable of classifying correctly that Andrew Scheer, Justin Trudeau, and Jagmeet Singh accounts are managed by humans.

Conclusions and Future Work

As described in this report, our work has achieved the goal of developing a cloud solution that uses the services best suited to the purpose of streaming data, storing it and making it available for viewing. The developed solution is capable of receiving a large volume of data, preprocessing and storing it for consumption. We successfully streamed over 189,000 tweets, applied data cleansing and sentiment analysis techniques to tweet content and user profile data.

The number of autonomous counts (bot) identified by Twitter is growing, and the activity of these accounts has an impact on the social structure and the way users interact. In this context, we performed logistic regression algorithm that was able to classify with an accuracy of 97% whether a user is a bot or not.

The solution was focused on the cloud platform, leaving room for future improvement in the analytics process. Towards Improving accuracy and avoiding overfitting, the training dataset, that currently has very few records, even after the applied SMOTE technique, needs to grow in volume. It is critical to the reliability of the result of the classification model that we are able to provide for it a larger sample of training data.

Another improvement to be made in the future is the inclusion of the cleanear cloud function directly in the stream pipeline. When receiving the message PubSub should call first the subscriber and then the cleansing function, currently what we do is schedule the cleaning for the end of the day.

To conclude, there are a variety of analyzes that can be performed to add value, such as trend topic analysis, assessing the impact of external events on tweet content sentiment, classify not only user profile but also tweet content. We believe that social media has a huge impact on the way we communicate, vote and consume, so every effort to understand this movement is valid.

References

Gaekwad, K. (2018, January 12). Welcome. Retrieved October 10, 2019, from https://www.lynda.com/Kubernetes-tutorials/Welcome/661764/693556-4.html.

Gilani, Z. & Farahbakhsh R. & Tyson, G. & Wang, L. & Crowcroft J. Of Bots and Humans (on Twitter). Retrieved November 30, 2019, from http://www-public.imtbs-tsp.eu/~farahbak/publications/Asonam_17.pdf

Glazer, E., & Moriarty, D. (2019, November 28). Presidential Candidates Take to Social Media. Retrieved November 30, 2019, from https://www.wsj.com/articles/presidential-candidates-take-to-social-media-11574942401

GoogleCloudPlatform. Google Cloud Platform. Retrieved September 15, 2019, from https://cloud.google.com/

GoogleCloudPlatform. GoogleCloudPlatform/kubernetes-bigquery-python. Retrieved September 3, 2019, from https://github.com/GoogleCloudPlatform/kubernetes-bigquery-python/tree/master/pubsub

Shaban, Hamza. "Twitter Reveals Its Daily Active User Numbers for the First Time." The Washington Post, WP Company, 7 Feb. 2019, https://www.washingtonpost.com/technology/2019/02/07/twitter-reveals-its-daily-active-user-numbers-first-time/

Text Analytics for Beginners using NLTK. (n.d.). Retrieved October 15, 2019, from https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk.

Tweepy. Retrieved August 15, 2019, from https://www.tweepy.org/

Twitter. Update on Twitter's review of the 2016 US election. (n.d.). Retrieved November 1, 2019, from https://blog.twitter.com/en_us/topics/company/2018/2016-election-update.html.

What is a Container? (n.d.). Retrieved October 10, 2019, from https://www.Docker.com/resources/what-container.

What is Kubernetes. (n.d.). Retrieved September 15, 2019, from https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/.

WOJCIK, S., & HUGHES, A. (2019, April 24). PEW Research Center. Sizing Up Twitter Users. Retrieved from https://www.pewresearch.org/internet/2019/04/24/sizing-up-twitter-users/

5 Things You Need to Know about Sentiment Analysis and Classification. (n.d.). Retrieved September 29, 2019, from https://www.kdnuggets.com/2018/03/5-things-sentiment-analysis-classification.html.